Logical Methods in Computer Science
Vol. 8 (1:28) 2012, pp. 1–44
www.lmcs-online.org

Submitted Oct. 4, 2011
Published Mar. 26, 2012

PRESERVATION OF STRONG NORMALISATION MODULO PERMUTATIONS FOR THE STRUCTURAL λ-CALCULUS

BENIAMINO ACCATTOLI [a] AND DELIA KESNER [b]

[a] INRIA Saclay and LIX (École Polytechnique)
e-mail address: beniamino.accattoli@gmail.com

[b] Univ. Paris Diderot, Sorbonne Paris Cité, PPS, CNRS
e-mail address: delia.kesner@pps.jussieu.fr

Abstract. Inspired by a recent graphical formalism for λ-calculus based on linear logic technology, we introduce an untyped structural λ-calculus, called $\lambda\mathtt{j}$, which combines actions at a distance with exponential rules decomposing the substitution by means of weakening, contraction and dereliction. First, we prove some fundamental properties of $\lambda\mathtt{j}$ such as confluence and preservation of β-strong normalisation. Second, we add a strong bisimulation to $\lambda\mathtt{j}$ by means of an equational theory which captures in particular Regnier's σ-equivalence. We then complete this bisimulation with two more equations for (de)composition of substitutions and we prove that the resulting calculus still preserves β-strong normalization. Finally, we discuss some consequences of our results.

Introduction

Linear Logic [13] has been very influential in computer science, especially because it provides a tool to explicitly control the use of resources by limiting the use of the *structural rules* of weakening and contraction. Erasure (weakening) and duplication (contraction) are restricted to formulas marked with an *exponential* modality, and can only interact with non-linear proofs marked with a *bang* modality. Intuitionistic and Classical Logic can thus be encoded by a fragment containing such modalities as, for example, the Multiplicative Exponential Linear Logic (MELL).

MELL proofs can be represented by sequent trees, but MELL Proof-Nets [13] provide a better geometrical representation of proofs, eliminating irrelevant syntactical details. They have been used extensively to develop different encodings of intuitionistic logic/lambda-calculus, giving rise to the geometry of interaction [14].

Normalisation of proofs (*i.e. cut elimination*) in MELL Proof-Nets is performed in particular by *exponential* and *commutative* rules. Non-linear proofs are distinguished by surrounding *boxes*; the exponential rules handle all the possible operations on them: erasure, duplication and linear replacement, corresponding respectively to a cut elimination step

1998 ACM Subject Classification: F.3.2, D.1.1, F.4.1.
Key words and phrases: Lambda-calculus, explicit substitutions, preservation of strong normalisation.

DOI:10.2168/LMCS-8 (1:28) 2012

involving a box and either a *weakening*, a *contraction* or a *dereliction*. The commutative rule instead *composes* non-linear resources.

Different cut elimination systems [11, 24, 20], defined as *explicit substitution* (ES) calculi, were explained in terms of, or were inspired from, the fine notion of reduction of MELL Proof-Nets. They all use the idea that the content of a substitution/cut is a non-linear resource, *i.e.* a box that can be composed with another one by means of some commutative rules. They also share common operational semantics defined in terms of a *propagation system* in which a substitution traverses a term until the variables are reached.

The structural λ-calculus. A graphical representation for λ-terms, $\lambda\mathtt{j}$-dags, has been recently proposed [2]. It denies boxes by representing them with additional edges called j*umps*, and does not need any commutative reduction rule to compose non-linear proofs. This paper studies the term formalism, called $\lambda\mathtt{j}$-calculus, resulting from reading back $\lambda\mathtt{j}$-dags (and their correspondent reductions) by means of their sequentialisation theorem [2]. The deep connection between $\lambda\mathtt{j}$-dags and Danos and Regnier's Pure (*untyped*) Proof-Nets [7] has been already studied in [1].

Beyond this graphical and logical interpretation, the peculiarity of $\lambda\mathtt{j}$-calculus is that it uses two features which were never combined before: *action at a distance* and *multiplicities*.

Action at a distance means that rewriting rules are specified by means of some constructors which are arbitrarily far away from each other. This approach could be understood as inconvenient but this is only apparent because rewriting rules can be locally implemented by means of $\lambda\mathtt{j}$-dags. The distance rules of $\lambda\mathtt{j}$ do not propagate substitutions through the term except for the linear ones which are evaluated exactly as meta-level substitutions, regardless the distance between the involved constructors (variable and jump).

Multiplicities are intended to count the number of occurrences of a given variable affected by a jump, *i.e.* the rewriting rule to be applied for reducing a term of the form $t[x/u]$ depends on $|t|_x$, the number of free occurrences of the variable x in the term t. Indeed, we distinguish three cases, $|t|_x = 0$, $|t|_x = 1$ and $|t|_x > 1$, which correspond, respectively, to weakening-box, dereliction-box and contraction-box cut-elimination rules in Proof Nets. It is because of the weakening and contraction rules that we call our language the *structural* λ-calculus.

Content of the paper. We start by showing that $\lambda\mathtt{j}$ admits a simple and elegant theory *i.e.* it enjoys confluence, full composition (FC), and preservation of β-strong normalisation (PSN). The proof of PSN is particularly concise because of the distance approach.

The main result of the paper is that the theory of $\lambda\mathtt{j}$ admits a modular extension with respect to propagations of jumps: an equational theory is added on top of $\lambda\mathtt{j}$ and the obtained extension is shown to preserve all the good properties we mentioned before. Actually, we focus on PSN, since FC and confluence for the extended $\lambda\mathtt{j}$-calculus result as straightforward.

In the literature there is a huge number of calculi with expicit substitutions, `let` constructs or environments, most of them use some rule to specify commutation (also called propagation or permutation). In order to encompass these formalisms we do not approach propagations as *rewriting rules*, but as equations (which can be used from left to right or vice-versa) defining an *equivalence relation* on terms.

This is only possible because propagations are not needed in $\lambda\mathtt{j}$ to compute normal forms, a fact which is a by-product of the *distance* notion. Moreover, any particular orientation of the equations (from left to right *or* from right to left) results in a terminating

rewriting relation, which implies that the system containing *any* orientation of the equations still enjoys PSN.

Equations are introduced in two steps. We first consider commutations between independent jumps and between jumps and abstractions or left sides of applications. This equivalence, written $\equiv_{\mathtt{o}}$, turns out to be a strong bisimulation, *i.e.* a reduction relation which is length preserving; thus PSN for the reduction system $\lambda\mathtt{j}$ modulo $\equiv_{\mathtt{o}}$ — noted $\lambda\mathtt{j/o}$ — immediately follows. We also show that $\equiv_{\mathtt{o}}$ can be seen as a projection of Regnier's σ-equivalence [37] on a syntax with jumps. Actually, $\equiv_{\mathtt{o}}$ can be understood as the quotient induced by the translation [1] of $\lambda\mathtt{j}$-terms to Pure Proof-Nets, which is why it is so well-behaved, and why we call it the *graphical equivalence*.

The second step is to extend $\equiv_{\mathtt{o}}$ with general commutations between jumps and right sides of applications and contents of jumps. The resulting *substitution equivalence* $\equiv_{\mathtt{obox}}$ does not only subsume *composition* of jumps, but also *decomposition*. The equations of $\equiv_{\mathtt{obox}}$ correspond exactly to the commutative box-box case of Proof-Nets, but they are here considered as an *equivalence* — which is a novelty — and not as a rewriting rule. The reduction relation of $\lambda\mathtt{j/obox}$ is a rich rewriting system with subtle behaviour, particularly because $\equiv_{\mathtt{obox}}$ affects reduction lengths, and thus is not a strong bisimulation. Nonetheless, we show that $\lambda\mathtt{j/obox}$ enjoys PSN.

This result is non-trivial, and constitutes the main contribution of the paper. The technique used to obtain PSN for $\lambda\mathtt{j/obox}$ consists in

(1) Projecting $\lambda\mathtt{j/obox}$ reductions into a calculus that we call $\lambda\mathtt{void/o}$,
(2) Proving PSN for $\lambda\mathtt{void/o}$,
(3) Infering PSN for $\lambda\mathtt{j/obox}$ from (1) and (2).

Actually, $\lambda\mathtt{void/o}$ can be understood as a *memory* calculus specified by means of *void* jumps — *i.e.* jumps $t[x/u]$ where $x \notin \mathtt{fv}(t)$ — which generalises Klop's Λ_I-calculus [27]. Despite the fact that it appears only as a technical tool we claim that it is a calculus interesting on its own and can be used for proving termination results beyond those of this paper.

The last part of the paper presents some interesting consequences of our main result concerning different variations on $\lambda\mathtt{j/obox}$.

Road Map.

- Section 1 recalls some general notions about abstract rewriting.
- Section 2 presents the $\lambda\mathtt{j}$-calculus and shows that it enjoys basic properties such as full composition, simulation of one-step β-reduction, and confluence.
- Section 3 studies preservation of β-strong normalisation (PSN). The PSN property is proved using a modular technique developed in [21], which results in a very short formal argument in our case.
- Section 4 first considers $\lambda\mathtt{j}$ enriched with the equivalence $\equiv_{\mathtt{o}}$, which is related to Regnier's σ-equivalence [37], and then with the equivalence $\equiv_{\mathtt{obox}}$, which also contains composition of jumps.
- Section 5 is devoted to the proof of PSN for $\lambda\mathtt{j}$ modulo $\equiv_{\mathtt{obox}}$, which is the main contribution of the paper.
- Section 6 discusses some consequences of the PSN result of Section 5.

This paper covers some basic results in [3] by extending them considerably. Indeed, the propagation systems considered in [3] are just particular cases of the general equational

theory $\equiv_{\texttt{obox}}$ studied in this paper. The proof technique used here to show PSN for $\lambda\texttt{j}$ modulo $\equiv_{\texttt{obox}}$ puts in evidence another calculus $\lambda\texttt{void/o}$ that has interest in itself. Moreover, interesting consequences of the main result are included in Section 6.

Related Work. Action at a distance has already been used in [33, 10, 35], but none of the previous approaches takes advantage of distance plus control of resources by means of multiplicities. Other works use multiplicities [25] but not distance so that the resulting formalism contains a lot of rules, which is really less manageable. We think that our combined approach is more primitive than ES, and the resulting theory is much simpler. Using distance *and* multiplicities also provides modularity: the substitution rules become independent from the set of constructors of the calculus, and thus any change in the language does not cause any changes in the associated rewriting rules. Our combined approach does not only capture the well-known notions of developments [17] and superdevelopments [28], but also allows us to introduce XL-developments, a more powerful notion of development defined in [3].

In the literature there are many calculi which dealt with permutations of constructors in intuitionistic calculi, but all use *reduction* rules rather than *equations*, which is less powerful. Some that can be captured by our graphical equivalence appear in [19, 37, 26] and those captured by our substitution equivalence are [12, 16, 43]. Intuitionistic calculi inspired from Linear Logic Proof Nets appear for example in [23, 21, 25].

1. Preliminary notions

As several reduction notions are used along the paper, we first introduce general definitions of rewriting.

A **reduction system** is a pair $(R, \to_{\mathcal{R}})$ consisting of a set R and a binary relation $\to_{\mathcal{R}}$ on R called a **reduction relation**. When $(a, b) \in \to_{\mathcal{R}}$ we write $a \to_{\mathcal{R}} b$ and we say that a **$\mathcal{R}$-reduces** to b. The inverse of $\to_{\mathcal{R}}$ is written ${}_{\mathcal{R}}\!\leftarrow$, *i.e.* $b\ {}_{\mathcal{R}}\!\leftarrow a$ iff $a \to_{\mathcal{R}} b$. The reflexive and transitive (resp. transitive) closure of $\to_{\mathcal{R}}$ is written $\to^*_{\mathcal{R}}$ (resp. $\to^+_{\mathcal{R}}$). Composition of relations is denoted by juxtaposition. Given $k \geq 0$, we write $a \overset{k}{\to}_{\mathcal{R}} b$ iff a is $\mathcal{R}$-related to b in k steps, *i.e.* $a \overset{0}{\to}_{\mathcal{R}} b$ if $a = b$ and $a \overset{n+1}{\to}_{\mathcal{R}} b$ if $\exists\, c$ s.t. $a \to_{\mathcal{R}} c$ and $c \overset{n}{\to}_{\mathcal{R}} b$.

Given a **reduction system** $(R, \to_{\mathcal{R}})$, we use the following reduction notions:

- $\mathcal{R}$ is **locally confluent** if ${}_{\mathcal{R}}\!\leftarrow \to_{\mathcal{R}} \subseteq \to^*_{\mathcal{R}}\ {}^*_{\mathcal{R}}\!\leftarrow$, *i.e.* if $a \to_{\mathcal{R}} b$ and $a \to_{\mathcal{R}} c$, then $\exists d$ s.t. $b \to^*_{\mathcal{R}} d$ and $c \to^*_{\mathcal{R}} d$.
- $\mathcal{R}$ is **confluent** if ${}^*_{\mathcal{R}}\!\leftarrow \to^*_{\mathcal{R}} \subseteq \to^*_{\mathcal{R}}\ {}^*_{\mathcal{R}}\!\leftarrow$, *i.e.* if $a \to^*_{\mathcal{R}} b$ and $a \to^*_{\mathcal{R}} c$, then $\exists d$ s.t. $b \to^*_{\mathcal{R}} d$ and $c \to^*_{\mathcal{R}} d$.
- $s \in R$ is in **$\mathcal{R}$-normal form**, written $s \in \mathcal{R}$**-nf**, if there is no s' such that $s \to_{\mathcal{R}} s'$.
- $s \in R$ has an **$\mathcal{R}$-normal form** iff there exists $u \in \mathcal{R}$-nf such that $s \to^*_{\mathcal{R}} u$. When s has a *unique* $\mathcal{R}$-normal form, this one is denoted by $\mathcal{R}(s)$.
- $s \in R$ is **$\mathcal{R}$-weakly normalizing**, written $s \in \mathcal{WN}_{\mathcal{R}}$, iff s has an $\mathcal{R}$-normal form.
- $s \in R$ is **$\mathcal{R}$-strongly normalizing** or **$\mathcal{R}$-terminating**, written $s \in \mathcal{SN}_{\mathcal{R}}$, if there is no infinite $\mathcal{R}$-reduction sequence starting at s.
- $s \in R$ is **$\mathcal{R}$-finitely branching** if the set $\{s' \mid s \to_{\mathcal{R}} s'\}$ is finite.
- If $s \in \mathcal{R}$ is $\mathcal{R}$-strongly normalizing and $\mathcal{R}$-finitely branching then $\eta_{\mathcal{R}}(s)$ denotes the **maximal length of an $\mathcal{R}$-reduction sequence starting at** s. This notion is extended to lists of terms by $\eta_{\mathcal{R}}(s_1 \dots s_m) = \sum_{i=1}^{m} \eta_{\mathcal{R}}(s_i)$.

- $\mathcal{R}$ is **weakly normalizing** (resp. **strongly normalizing** or **terminating**) if every $s \in \mathcal{R}$ is.

A **strong bisimulation between** two reduction systems $(S, \to_S)$ and $(Q, \to_\mathcal{Q})$ is a relation $\mathtt{E} \subseteq S \times Q$ s.t. for any pair $s \mathtt{E} t$:

- If $s \to_\mathcal{S} s'$ then there is t' s.t. $t \to_\mathcal{Q} t'$ and $s' \mathtt{E} t'$, and conversely:
- If $t \to_\mathcal{Q} t'$ then there is s' s.t. $s \to_\mathcal{S} s'$ and $s' \mathtt{E} t'$.

A **strong bisimulation for** $(S, \to_S)$ is a strong bisimulation between $(S, \to_S)$ and itself. In particular we shall make use of the following property whose proof is straightforward:

Lemma 1.1. *Let* $\mathtt{E}$ *be a strong bisimulation between two reduction systems* $(S, \to_\mathcal{S})$ *and* $(Q, \to_\mathcal{Q})$.

(1) The relation $\mathtt{E}$ *preserves reduction lengths, i.e. for any* $s \mathtt{E} t$
- *If* $s \to_\mathcal{S}^k s'$ *then* $\exists\, t'$ *s.t.* $t \to_\mathcal{Q}^k t'$ *and* $s' \mathtt{E} t'$.
- *If* $t \to_\mathcal{Q}^k t'$ *then* $\exists\, s'$ *s.t.* $s \to_\mathcal{S}^k s'$ *and* $s' \mathtt{E} t'$.

(2) The relation $\mathtt{E}$ *preserves strong normalization, i.e. for any* $s \mathtt{E} t, s \in \mathcal{SN}_\mathcal{S}$ *if and only if* $t \in \mathcal{SN}_\mathcal{Q}$.

Given a reduction relation $\to_\mathcal{S}$ and an equivalence relation $\mathtt{E}$ both on S, the reduction relation $\to_{\mathcal{S}/\mathtt{E}}$, called **reduction** $\mathcal{S}$ **modulo** $\mathtt{E}$, is defined by $t \to_{\mathcal{S}/\mathtt{E}} u$ iff $t \mathtt{E} t' \to_\mathcal{S} u' \mathtt{E} u$.

Lemma 1.2. *Let* $\mathtt{E}$ *be a strong bisimulation for* $(S, \to_\mathcal{S})$. *Then,*

(1) The relation $\mathtt{E}$ *can be postponed w.r.t* $\to_\mathcal{S}$, *i.e.* $\to_{\mathcal{S}/\mathtt{E}}^* = \to_\mathcal{S}^* \mathtt{E}$.

(2) If $\to_\mathcal{S}$ *is confluent then* $\to_{\mathcal{S}/\mathtt{E}}$ *is confluent.*

(3) If $t \in \mathcal{SN}_\mathcal{S}$, *then* $t \in \mathcal{SN}_{\mathcal{S}/\mathtt{E}}$.

Proof. Point 1 is straightforward by induction on the length of $\to_{\mathcal{S}/\mathtt{E}}^*$ using the definition of strong bisimulation. Points 2 and 3 follow from Point 1. □

We conclude this section by giving an abstract theorem that we will use to prove strong normalisation for different notions of reduction modulo.

Theorem 1.3 (Termination for reduction modulo by interpretation)**.** *Let consider three reduction systems* $(A, \to_{\mathcal{A}_1})$, $(A, \to_{\mathcal{A}_2})$ *and* $(B, \to_\mathcal{B})$. *Let* $\mathtt{E}$ *(resp.* $\mathtt{F}$*) be an equivalence on* A *(resp.* B*). Consider a relation* $\mathtt{R} \subseteq A \times B$. *Suppose that for all* u, v, U

(P0) $u \mathtt{R} U$ & $u \mathtt{E} v$ *imply* $\exists V$ *s.t.* $v \mathtt{R} V$ & $U \mathtt{F} V$.
(P1) $u \mathtt{R} U$ & $u \to_{\mathcal{A}_1} v$ *imply* $\exists V$ *s.t.* $v \mathtt{R} V$ & $U \to_\mathcal{B}^* V$.
(P2) $u \mathtt{R} U$ & $u \to_{\mathcal{A}_2} v$ *imply* $\exists V$ *s.t.* $v \mathtt{R} V$ & $U \to_\mathcal{B}^+ V$.
(P3) *The reduction relation* $\to_{\mathcal{A}_1/\mathtt{E}}$ *is terminating.*

Then, $t \mathtt{R} T$ & $T \in \mathcal{SN}_{\mathcal{B}/\mathtt{F}}$ *imply* $t \in \mathcal{SN}_{(\mathcal{A}_1 \cup \mathcal{A}_2)/\mathtt{E}}$.

Proof. Suppose $t \notin \mathcal{SN}_{(\mathcal{A}_1 \cup \mathcal{A}_2)/\mathtt{E}}$. Then, there is an infinite $(\mathcal{A}_1 \cup \mathcal{A}_2)/\mathtt{E}$-reduction sequence starting at t, and since $\to_{\mathcal{A}_1/\mathtt{E}}$ is a terminating reduction relation by $(P3)$, this reduction has necessarily the form:

$$t \;\to_{\mathcal{A}_1/\mathtt{E}}^*\; t_1 \;\to_{\mathcal{A}_2/\mathtt{E}}^+\; t_2 \;\to_{\mathcal{A}_1/\mathtt{E}}^*\; t_3 \;\to_{\mathcal{A}_2/\mathtt{E}}^+\; t_4 \;\to_{\mathcal{A}_1/\mathtt{E}}^* \cdots$$

And can be projected by $(P0)$, $(P1)$ and $(P2)$ into an infinite $\mathcal{B}$ reduction sequence as follows:

$$\begin{array}{ccccccccccc} t & \to^*_{\mathcal{A}_1/\mathtt{E}} & t_1 & \to^+_{\mathcal{A}_2/\mathtt{E}} & t_2 & \to^*_{\mathcal{A}_1/\mathtt{E}} & t_3 & \to^+_{\mathcal{A}_2/\mathtt{E}} & t_4 & \to^*_{\mathcal{A}_1/\mathtt{E}} \cdots \\ \mathtt{R} & & \mathtt{R} & & \mathtt{R} & & \mathtt{R} & & \mathtt{R} & \\ T & \to^*_{\mathcal{B}/\mathtt{F}} & T_1 & \to^+_{\mathcal{B}/\mathtt{F}} & T_2 & \to^*_{\mathcal{B}/\mathtt{F}} & T_3 & \to^+_{\mathcal{B}/\mathtt{F}} & T_4 & \to^*_{\mathcal{B}/\mathtt{F}} \cdots \end{array}$$

Since $T \in \mathcal{SN}_{\mathcal{B}/\mathtt{F}}$, then we get a contradiction. □

2. The structural λj-calculus

We introduce in this section the structural $\lambda\mathtt{j}$-calculus, which can simply be understood as a refinement of λ-calculus. To be self-contained, we start this section by recalling the syntax and semantics of λ-calculus. The set of λ-terms, written $\mathcal{T}_\lambda$, is generated by the following grammar:

$$(\mathcal{T}_\lambda) \quad t, u ::= x \mid \lambda x.t \mid tu$$

Dynamics of λ-terms is given by β-reduction (noted $\to_\beta$) which is defined as the closure by contexts of the following reduction rule:

$$(\lambda x.t)u \mapsto_\beta t\{x/u\}$$

where the meta-operation $t\{x/u\}$ on λ-terms is just a particular case of the meta-operation on $\lambda\mathtt{j}$-terms given below.

The **structural $\lambda\mathtt{j}$-calculus** is given by a set of terms and a set of reduction rules. The set of $\lambda\mathtt{j}$-terms, written $\mathcal{T}$, is generated by the following grammar:

$$(\mathcal{T}) \quad t, u ::= x \mid \lambda x.t \mid tu \mid t[x/u]$$

The term x is **variable**, $\lambda x.t$ an **abstraction**, tu an **application** and $t[x/u]$ a **substituted term**. The object $[x/u]$, which is not a term, is called a **jump**. The terms $\lambda x.t$ and $t[x/u]$ bind x in t, *i.e.* the sets of **free/bound variables** of a term are given by the following definitions:

$$\begin{array}{lcllcl} \mathtt{fv}(x) & := & \{x\} & \mathtt{bv}(x) & := & \emptyset \\ \mathtt{fv}(tu) & := & \mathtt{fv}(t) \cup \mathtt{fv}(u) & \mathtt{bv}(tu) & := & \mathtt{bv}(t) \cup \mathtt{bv}(u) \\ \mathtt{fv}(\lambda x.t) & := & \mathtt{fv}(t) \setminus \{x\} & \mathtt{bv}(\lambda x.t) & := & \mathtt{bv}(t) \cup \{x\} \\ \mathtt{fv}(t[x/u]) & := & (\mathtt{fv}(t) \setminus \{x\}) \cup \mathtt{fv}(u) & \mathtt{bv}(t[x/u]) & := & \mathtt{bv}(t) \cup \{x\} \cup \mathtt{bv}(u) \end{array}$$

A jump $[x/u]$ in a term $t[x/u]$ is called **void** if $x \notin \mathtt{fv}(t)$. The equivalence relation generated by the renaming of bound variables is called **α-conversion**. Thus for example $(\lambda y.x)[x/y] \equiv_\alpha (\lambda y'.x')[x'/y]$. The notation $\overline{t}^1_n$ is used for the empty sequence of terms if $n = 0$ and for the sequence $[t_1; \ldots; t_n]$ otherwise; $\overline{t}^1_n \subseteq S$ means that all the elements of the sequence belong to the set S. If $i, n \in \mathbb{N}$ we use $v\overline{t}^i_n$ for the term v if $n < i$ and $(vt_i)\overline{t}^{i+1}_n$ otherwise; similarly, $t[x_i/u_i]^i_n$ denotes the term t if $n < i$ and $t[x_i/u_i][x_i/u_i]^{i+1}_n$ otherwise; $t_1 t_2 \ldots t_n$ $(n \geq 1)$ denotes the application $(\ldots(t_1 t_2)\ldots\ldots)t_n$;

The meta-level **substitution** operation is defined by induction on terms by using the following equations on α-equivalence classes:

$$\begin{array}{lll} x\{x/u\} & := u & \\ y\{x/u\} & := y & \\ (\lambda y.t)\{x/u\} & := \lambda y.t\{x/u\} & \text{if } y \notin \mathtt{fv}(u) \\ (tv)\{x/u\} & := t\{x/u\}v\{x/u\} & \\ t[y/v]\{x/u\} & := t\{x/u\}[y/v\{x/u\}] & \text{if } y \notin \mathtt{fv}(u) \end{array}$$

We write $t \triangleright u$ or $u \triangleleft t$ when u is a (strict) subterm of t. **Positions** of terms are defined as expected (see [42], p. 643, for details); $t|_p$ denotes the **subterm of** t **at position** p and $\mathtt{pos}_x(t)$ denotes the **set of all the positions** p of t s.t. $t|_p = x$.

We use $|t|$ to denote the size of t. We write $|t|_x$ for the number of free occurrences of the variable x in the term t, called the **multiplicy of** x **in** t. We extend this notion to sets of variables by $|t|_\Gamma := \Sigma_{x\in\Gamma}|t|_x$. A key notion used to define the semantics of the $\lambda\mathtt{j}$-calculus is that of renaming: given a term t and a subset $S \subseteq \mathtt{pos}_x(t) \cap \mathtt{fv}(t)$, we write $R_y^{S,x}(t)$ for the term t' verifying $t'|_p = t|_p$ if $p \notin \mathcal{S}$ and $t'|_p = y$ if $(t|_p = x \ \&\ p \in S)$. Thus for example, $R_y^{\{111,2\},x}(xzxx) = yzxy$.

When $|t|_x = n \geq 2$, we write $t_{[y]_x}$ for any **non-deterministic replacement** of i $(1 \leq i \leq n-1)$ occurrences of x in t by a *fresh* variable y, *i.e.* $t_{[y]_x}$ denotes any term $R_y^{S,x}(t)$ s.t. $|S| \geq 2$ and $S \subset \mathtt{pos}_x(t)$. Thus for example, $(xxxx)_{[y]_x}$ may denote $(yxyx)$ or $(xyyy)$ but not $(yyyy)$.

Contexts are generated by the following grammar:

$$C ::= \Box \mid Cv \mid vC \mid v[y/C] \mid C[y/v] \mid \lambda y.C$$

We write $C[\![t]\!]$ to denote the term obtained by replacing the hole $\Box$ in C by the term t. Thus for example $\lambda x.z[y/w\Box][\![x]\!] = \lambda x.z[y/wx]$ (remark that capture of variables is possible).

The **binding set** of a context is defined as follows:

$$\begin{array}{lllllll} \mathtt{bs}(\Box) & := & \emptyset & \mathtt{bs}(t[x/C]) & := & \mathtt{bs}(C) \\ \mathtt{bs}(tC) & := & \mathtt{bs}(C) & \mathtt{bs}(C[x/v]) & := & \mathtt{bs}(C) \cup \{x\} \\ \mathtt{bs}(Cv) & := & \mathtt{bs}(C) & \mathtt{bs}(\lambda x.C) & := & \mathtt{bs}(C) \cup \{x\} \end{array}$$

We now consider the rewriting rules of the structural λ-calculus (Figure 1), which decompose the β-rule into a finer set of rules. The letter $\mathtt{L}$ in the rule $\mathtt{dB}$ denotes a list $[x_1/u_1]\dots[x_k/u_k]$ of jumps with $k \in \mathbb{N}$ (so potentially $k = 0$) such that $\{x_1,\dots,x_k\} \cap \mathtt{fv}(u) = \emptyset$. The $\mathtt{dB}$ rule extends the usual $\mathtt{B}$ rule $(\lambda x.t)u \to_\mathtt{B} t[x/u]$ by allowing to introduce some distance between the abstraction $\lambda x.t$ and the argument u which is specified by means of a list of substitutions $\mathtt{L}$. This natural extension comes from reading back a multiplicative cut in $\lambda\mathtt{j}$-dags or Pure Proof-Nets [2, 1].

The substitution rules also deserve some explanation. The side conditions $|t|_x = 0$, $|t|_x = 1$ and $|t|_x > 1$ are global on terms but local on graphs, simply because in the graph all the occurrences of the same variable are grouped together. Also, the (global) meta-substitution operation $t\{x/u\}$ used in the right-hand side of the rule $\mathtt{d}$ is completely local on graphs. Similarly, the meta-operation $t_{[y]_x}$ used in the right-hand side of the $\mathtt{c}$-rule is an algebraic notation for the local operation on graphs which splits the co-located occurrences of x into two disjoint and non-empty sets, one of which corresponds to x, while the other is associated to the fresh variable y. Thus, the structural λ-calculus can be seen as an algebraic language useful to study $\lambda\mathtt{j}$-dags and Pure Proof-Nets.

$$
\begin{array}{llll}
\text{(Beta at a distance)} & (\lambda x.t)\mathtt{L}\ u \mapsto_{\mathtt{dB}} & t[x/u]\mathtt{L} & \\
\text{(weakening)} & t[x/u] \mapsto_{\mathtt{w}} & t & \text{if } |t|_x = 0 \\
\text{(dereliction)} & t[x/u] \mapsto_{\mathtt{d}} & t\{x/u\} & \text{if } |t|_x = 1 \\
\text{(contraction)} & t[x/u] \mapsto_{\mathtt{c}} & t_{[y]_x}[x/u][y/u] & \text{if } |t|_x > 1
\end{array}
$$

Figure 1: The $\lambda\mathtt{j}$-reduction system

We close these rules by contexts, as usual: $\to_{\mathcal{R}}$ denotes the contextual closure of $\mapsto_{\mathcal{R}}$, for $\mathcal{R} \subseteq \{\mathtt{dB}, \mathtt{w}, \mathtt{d}, \mathtt{c}\}$. We write $\to_{\neg\mathtt{w}}$ for the reduction relation $\to_{\mathtt{dB},\mathtt{d},\mathtt{c}}$. The **reduction relation** $\to_{\lambda\mathtt{j}}$ (resp. $\to_{\mathtt{j}}$) is generated by all (resp. all expect $\mathtt{dB}$) the previous rewriting rules modulo α-conversion.

An expected property of $\lambda\mathtt{j}$ is that the reduction relation $\lambda\mathtt{j}$ is stable by substitution.

Lemma 2.1. *Let $t, u \in$ terms.*
- *If $t \to_{\lambda\mathtt{j}} t'$, then $t\{x/u\} \to_{\lambda\mathtt{j}} t'\{x/u\}$.*
- *If $u \to_{\lambda\mathtt{j}} u'$, then $t\{x/u\} \to^*_{\lambda\mathtt{j}} t\{x/u'\}$.*

In the rest of this section we shall prove the following properties of $\lambda\mathtt{j}$: full composition (Lemma 2.2), simulation of one step β-reduction (Lemma 2.4), termination and uniqueness of normal forms of the substitution calculus $\to_{\mathtt{j}}$ (Lemmas 2.9 and 2.10), postponement of erasing reductions (Lemma 2.12) and confluence of $\lambda\mathtt{j}$ (Theorem 2.16).

2.1. **Jumps and Multiplicities.** The first property we show in this section is full composition, stating that any jump $[x/u]$ in a substituted term $t[x/u]$ can be reduced to its implicit form $t\{x/u\}$. There are two interesting points. The first is that in contrast with most calculi of explicit substitutions, full composition holds with no need of equivalences. The second is that the proof is by induction on $|t|_x$ and not on the structure of t.

Lemma 2.2 (Full Composition (FC))**.** *Let $t, u \in \mathcal{T}$. Then $t[x/u] \to^+_{\mathtt{j}} t\{x/u\}$. Moreover, $|t|_x \geq 1$ implies $t[x/u] \to^+_{\mathtt{d},\mathtt{c}} t\{x/u\}$.*

Proof. By induction on $|t|_x$.
- If $|t|_x = 0$, then $t[x/u] \to_{\mathtt{w}} t = t\{x/u\}$.
- If $|t|_x = 1$, then $t[x/u] \to_{\mathtt{d}} t\{x/u\}$.
- If $|t|_x \geq 2$, then

$$
\begin{array}{lllll}
t[x/u] & \to_{\mathtt{c}} & t_{[y]_x}[y/u][x/u] & \to^+_{\mathtt{j}} & (i.h.) \\
 & & t_{[y]_x}\{y/u\}[x/u] & \to^+_{\mathtt{j}} & (i.h.) \\
 & & t_{[y]_x}\{y/u\}\{x/u\} & = & t\{x/u\}
\end{array}
$$

□

Due to the very general form of the duplication rule of $\lambda\mathtt{j}$, we get the following corollary which together with full composition can be seen as a generalised composition property:

Corollary 2.3. *Given $S \subset \mathtt{pos}_x(t)$ s.t. $|S| \geq 2$, then $t[x/u] \to^+_{\mathtt{j}} R_y^{S,x}(t)\{y/u\}[x/u]$, where y is a fresh variable.*

Proof. The term $t[x/u]$ $\mathtt{c}$-reduces to $R_y^{S,x}(t)[y/u][x/u]$. We conclude by full composition. □

Thus for example $(x(xx))[x/u] \to^+_{\lambda \mathtt{j}} (x(ux))[x/u]$. Note that this property is not enjoyed by traditional explicit substitution calculi: for instance, in $\lambda\mathtt{x}$ [6], the term $(x(xx))[x/u]$ cannot be reduced to $(x(ux))[x/u]$. However, it holds in calculi with partial substitutions, as Milner's calculus $\lambda\mathtt{sub}$ [33]. It is not difficult (see *e.g.* [22]) to define a translation $\mathtt{T}$ on terms such that $t \to_{\lambda\mathtt{sub}} t'$ implies $\mathtt{T}(t) \to^+_{\lambda \mathtt{j}} \mathtt{T}(t')$. This property allows in particular to deduce normalisation properties for $\lambda\mathtt{sub}$ from those of $\lambda\mathtt{j}$.

The one-step simulation of λ-calculus follows directly from full composition:

Lemma 2.4 (Simulation of λ-calculus)**.** *Let $t \in \mathcal{T}_\lambda$. If $t \to_\beta t'$ then $t \to^+_{\lambda \mathtt{j}} t'$.*

Proof. By induction on $t \to_\beta t'$. Let $t = (\lambda x.u)v \to_\beta u\{x/v\}$, then $t \to_{\mathtt{dB}} u[x/v] \to^+_{\mathtt{j}}$ (*Lem.* 2.2) $u\{x/v\}$. All the other cases are straightforward. □

We now introduce a notion that will be useful in various proofs. It counts the maximal number of free occurrences of a variable x that may appear during a $\mathtt{j}$-reduction sequence from a term t.

The **potential multiplicity** of the variable x in the term t, written $\mathtt{P}_x(t)$, is defined on α-equivalence classes as follows: if $x \notin \mathtt{fv}(t)$, then $\mathtt{P}_x(t) := 0$; otherwise:

$$\begin{array}{lcl} \mathtt{P}_x(x) & := & 1 \\ \mathtt{P}_x(\lambda y.u) & := & \mathtt{P}_x(u) \\ \mathtt{P}_x(uv) & := & \mathtt{P}_x(u) + \mathtt{P}_x(v) \\ \mathtt{P}_x(u[y/v]) & := & \mathtt{P}_x(u) + \mathtt{max}(1, \mathtt{P}_y(u)) \cdot \mathtt{P}_x(v) \end{array}$$

We can formalise the intuition behind $\mathtt{P}_x(t)$ as follows.

Lemma 2.5. *Let $t \in \mathcal{T}$. Then*
(1) $|t|_x \leq \mathtt{P}_x(t)$.
(2) *If t is a $\mathtt{c}$-nf then $|t|_x = \mathtt{P}_x(t)$.*

Proof. Both points are by induction on the definition of $\mathtt{P}_x(t)$. The only interesting case is when $t = u[y/v]$: the i.h. gives $|u|_x \leq \mathtt{P}_x(u)$, $|u|_y \leq \mathtt{P}_y(u)$ and $|v|_x \leq \mathtt{P}_x(v)$, from which we conclude with the first point. For the second one, if t is a $\mathtt{c}$-nf every relation given by the i.h. is an equality and $|u|_y = \mathtt{P}_y(u) \leq 1$, otherwise there would be a $\mathtt{c}$-redex. Then we get $\mathtt{P}_x(t) = \mathtt{P}_x(u) + \mathtt{max}(1, \mathtt{P}_y(u)) \cdot \mathtt{P}_x(v) = |u|_x + |v|_x = |t|_x$. □

Potential multiplicities enjoy the following properties.

Lemma 2.6. *Let $t \in \mathcal{T}$. Let x, y, z be pairwise distinct variables.*
(1) *If $u \in \mathcal{T}$ and $y \notin \mathtt{fv}(u)$, then $\mathtt{P}_y(t) = \mathtt{P}_y(t\{x/u\})$.*
(2) *If $|t|_x \geq 2$, then $\mathtt{P}_z(t) = \mathtt{P}_z(t_{[y]_x})$ and $\mathtt{P}_x(t) = \mathtt{P}_x(t_{[y]_x}) + \mathtt{P}_y(t_{[y]_x})$, where the two occurrences of the term $t_{[y]_x}$ denote exactly the same term.*
(3) *If $t \to_{\mathtt{j}} t'$, then $\mathtt{P}_y(t) \geq \mathtt{P}_y(t')$.*

Proof. By induction on t. □

By exploiting potential multiplicities we can define a measure of the global degree of sharing of a given term, and use this measure to prove that the j-reduction subsystem terminates.

We consider multisets of integers. We use $\emptyset$ to denote the empty multiset, $\sqcup$ to denote multiset union, and $\sqsupseteq$ (resp. $\sqsupset$) for the standard order (resp. strict order) on multisets [5]. Given an integer n and a multiset M, $n \cdot M$ denotes $\emptyset$ if $M = \emptyset$ and the multiset $[n \cdot a_1, \ldots, n \cdot a_n]$ if $M = [a_1, \ldots, a_n]$. The **j-measure** of $t \in \mathcal{T}$, written $\mathtt{jm}(t)$, is given by:

$$\begin{array}{lcl} \mathtt{jm}(x) & := & \emptyset \\ \mathtt{jm}(\lambda x.u) & := & \mathtt{jm}(u) \\ \mathtt{jm}(uv) & := & \mathtt{jm}(u) \sqcup \mathtt{jm}(v) \\ \mathtt{jm}(u[x/v]) & := & [\mathtt{P}_x(u)] \sqcup \mathtt{jm}(u) \sqcup \mathtt{max}(1, \mathtt{P}_x(u)) \cdot \mathtt{jm}(v) \end{array}$$

Note that $\mathtt{jm}(u) = \emptyset$ for $u \in \mathcal{T}_\lambda$. Potential multiplicities are decreasing by j-reduction, and we are going to show that the j-measure is strictly decreasing; however both can be incremented by dB-steps. For example, consider $t = (\lambda x.xx)y \to_{\mathtt{dB}} (xx)[x/y] = t'$. We get $\mathtt{P}_y(t) = 1$, $\mathtt{P}_y(t') = 2$, $\mathtt{jm}(t) = \emptyset$ and $\mathtt{jm}(t') = [2]$.

The fact that the j-measure decreases by j-reduction is proved as follows:

Lemma 2.7. *Let $t \in \mathcal{T}$. Then,*

(1) $\mathtt{jm}(t) = \mathtt{jm}(t_{[y]_x})$.
(2) *If $|t|_x = 1$, then* $\mathtt{jm}(t[x/u]) \sqsupset \mathtt{jm}(t\{x/u\})$.

Proof. By induction on t. The proof of the first property is straightforward. For the second one we show $[\mathtt{P}_x(t)] \sqcup \mathtt{jm}(t) \sqcup \mathtt{max}(1, \mathtt{P}_x(t)) \cdot \mathtt{jm}(u) \sqsupset \mathtt{jm}(t\{x/u\})$, which proves the desired property.

- $t = x$. Then $[1] \sqcup \mathtt{jm}(u) \sqsupset \mathtt{jm}(u) = \mathtt{jm}(x\{x/u\})$.
- $t = t_1[y/t_2]$. W.l.g we assume $y \notin \mathtt{fv}(u)$.

  If $x \in \mathtt{fv}(t_1)$, we reason as follows:

$$\begin{array}{ll} [\mathtt{P}_x(t)] \sqcup \mathtt{jm}(t) \sqcup \mathtt{max}(1, \mathtt{P}_x(t)) \cdot \mathtt{jm}(u) & = \\ [\mathtt{P}_x(t_1)] \sqcup [\mathtt{P}_y(t_1)] \sqcup \mathtt{jm}(t_1) \sqcup \mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{jm}(t_2) \sqcup \mathtt{max}(1, \mathtt{P}_x(t_1)) \cdot \mathtt{jm}(u) & \sqsupset_{i.h.} \\ [\mathtt{P}_y(t_1)] \sqcup \mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{jm}(t_2) \sqcup \mathtt{jm}(t_1\{x/u\}) & =_{Lem.\ 2.6:1} \\ [\mathtt{P}_y(t_1\{x/u\})] \sqcup \mathtt{max}(1, \mathtt{P}_y(t_1\{x/u\})) \cdot \mathtt{jm}(t_2) \sqcup \mathtt{jm}(t_1\{x/u\}) & = \\ \mathtt{jm}(t_1\{x/u\}[y/t_2]) & = \\ \mathtt{jm}(t\{x/u\}) & \end{array}$$

  If $x \in \mathtt{fv}(t_2)$, then $1 \leq \mathtt{P}_x(t_2)$ by Lemma 2.5:1 and so $\mathtt{max}(1, \mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{P}_x(t_2)) = \mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{P}_x(t_2) = \mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{max}(1, \mathtt{P}_x(t_2))$. Therefore:

$$\begin{array}{ll} [\mathtt{P}_x(t)] \sqcup \mathtt{jm}(t) \sqcup \mathtt{max}(1, \mathtt{P}_x(t)) \cdot \mathtt{jm}(u) & = \\ [\mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{P}_x(t_2)] \sqcup [\mathtt{P}_y(t_1)] \sqcup \mathtt{jm}(t_1) \sqcup \mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{jm}(t_2) & \\ \sqcup\ \mathtt{max}(1, \mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{P}_x(t_2)) \cdot \mathtt{jm}(u) & = \\ [\mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{P}_x(t_2)] \sqcup [\mathtt{P}_y(t_1)] \sqcup \mathtt{jm}(t_1) \sqcup \mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{jm}(t_2) & \\ \sqcup\ \mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot \mathtt{max}(1, \mathtt{P}_x(t_2)) \cdot \mathtt{jm}(u) & = \\ [\mathtt{P}_y(t_1)] \sqcup \mathtt{jm}(t_1)\ \sqcup \mathtt{max}(1, \mathtt{P}_y(t_1)) \cdot ([\mathtt{P}_x(t_2)] \sqcup \mathtt{jm}(t_2) \sqcup \mathtt{max}(1, \mathtt{P}_x(t_2)) \cdot \mathtt{jm}(u)) & \sqsupset_{i.h.} \\ [\mathtt{P}_y(t_1)] \sqcup \mathtt{jm}(t_1) \sqcup \mathtt{jm}(t_2\{x/u\}) & = \\ \mathtt{jm}(t\{x/u\}) & \end{array}$$

- All the other cases are straightforward. □

Lemma 2.8. *Let $t_0 \in \mathcal{T}$. Then,*
(1) $t_0 \equiv_\alpha t_1$ *implies* $\mathtt{jm}(t_0) = \mathtt{jm}(t_1)$.
(2) $t_0 \to_{\mathtt{j}} t_1$ *implies* $\mathtt{jm}(t_0) \sqsupset \mathtt{jm}(t_1)$.

Proof. By induction on the relations. The first point is straightforward, hence we only show the second one. We reason by cases.

- $t_0 = t[x/u] \to_{\mathtt{w}} t = t_1$, with $|t|_x = 0$. Then $\mathtt{jm}(t_0) = \mathtt{jm}(t) \sqcup 1 \cdot \mathtt{jm}(u) \sqcup [0] \sqsupset \mathtt{jm}(t) = \mathtt{jm}(t_1)$.
- $t_0 = t[x/u] \to_{\mathtt{d}} t\{x/u\} = t_1$, with $|t|_x = 1$. Then $\mathtt{jm}(t[x/u]) \sqsupset_{Lem.\ 2.7:2} \mathtt{jm}(t\{x/u\})$.
- $t_0 = t[x/u] \to_{\mathtt{c}} t_{[y]_x}[x/u][y/u] = t_1$, with $|t|_x \geq 2$ and y fresh. Then, Lemma 2.6:2 gives $[\mathtt{P}_x(t)] \sqsupset [\mathtt{P}_x(t_{[y]_x})] \sqcup [\mathtt{P}_y(t_{[y]_x})]$ and thus:

$$\begin{array}{ll}
\mathtt{jm}(t_0) & = \\
[\mathtt{P}_x(t)] \sqcup \mathtt{jm}(t) \sqcup \mathtt{P}_x(t) \cdot \mathtt{jm}(u) & = \\
[\mathtt{P}_x(t)] \sqcup \mathtt{jm}(t) \sqcup (\mathtt{P}_x(t_{[y]_x}) + \mathtt{P}_y(t_{[y]_x})) \cdot \mathtt{jm}(u) & =_{Lem.2.7:1} \\
[\mathtt{P}_x(t)] \sqcup \mathtt{jm}(t_{[y]_x}) \sqcup (\mathtt{P}_x(t_{[y]_x}) + \mathtt{P}_y(t_{[y]_x})) \cdot \mathtt{jm}(u) & \sqsupset_{Lem.\ 2.6:2} \\
[\mathtt{P}_x(t_{[y]_x})] \sqcup [\mathtt{P}_y(t_{[y]_x})] \sqcup \mathtt{jm}(t_{[y]_x}) \sqcup \mathtt{P}_x(t_{[y]_x}) \cdot \mathtt{jm}(u) \sqcup \mathtt{P}_y(t_{[y]_x}) \cdot \mathtt{jm}(u) & = \\
[\mathtt{P}_x(t_{[y]_x})] \sqcup [\mathtt{P}_y(t_{[y]_x}[x/u])] \sqcup \mathtt{jm}(t_{[y]_x}) \sqcup \mathtt{P}_x(t_{[y]_x}) \cdot \mathtt{jm}(u) \sqcup \mathtt{P}_y(t_{[y]_x}[x/u]) \cdot \mathtt{jm}(u) & = \\
[\mathtt{P}_y(t_{[y]_x}[x/u])] \sqcup \mathtt{jm}(t_{[y]_x}[x/u]) \sqcup \mathtt{P}_y(t_{[y]_x}[x/u]) \cdot \mathtt{jm}(u) & = \mathtt{jm}(t_1)
\end{array}$$

- $t_0 = t[x/u] \to t'[x/u] = t_1$, where $t \to t'$. Then:

$$\begin{array}{rcll}
\mathtt{jm}(t_0) & = & [\mathtt{P}_x(t)] \sqcup \mathtt{jm}(t) \sqcup \mathtt{max}(1, \mathtt{P}_x(t)) \cdot \mathtt{jm}(u) & \sqsupset_{i.h.} \\
 & & [\mathtt{P}_x(t)] \sqcup \mathtt{jm}(t') \sqcup \mathtt{max}(1, \mathtt{P}_x(t)) \cdot \mathtt{jm}(u) & \sqsupset_{Lem.\ 2.6:3} \\
 & & [\mathtt{P}_x(t')] \sqcup \mathtt{jm}(t') \sqcup \mathtt{max}(1, \mathtt{P}_x(t')) \cdot \mathtt{jm}(u) & = \mathtt{jm}(t_1)
\end{array}$$

- All the other cases are straightforward. □

The last lemma obviously implies:

Lemma 2.9. *The* $\mathtt{j}$*-calculus terminates.*

Furthermore:

Lemma 2.10. *The* $\mathtt{j}$*-reduction relation is confluent and terminating. Moreover, if* $\mathtt{j}(t)$ *denotes the (unique)* $\mathtt{j}$*-normal form of* t*, then the following properties hold:*

$$\begin{array}{rclcrcl}
\mathtt{j}(x) & = & x & \qquad & \mathtt{j}(uv) & = & \mathtt{j}(u)\mathtt{j}(v) \\
\mathtt{j}(\lambda x.u) & = & \lambda x.\mathtt{j}(u) & & \mathtt{j}(u[x/v]) & = & \mathtt{j}(u)\{x/\mathtt{j}(v)\}
\end{array}$$

Proof. One easily shows that $\to_{\mathtt{j}}$ is locally confluent, then Lemma 2.9 allows to apply Newman's Lemma [42] to conclude with the first part of the statament. The second part can be shown by induction on the structure of terms. Particularly, when $t = u[x/v]$ one has $u[x/v] \to_{\mathtt{j}}^* \mathtt{j}(u)[x/\mathtt{j}(v)] \to_{\mathtt{j}}^+$ (*Lem.* 2.2) $\mathtt{j}(u)\{x/\mathtt{j}(v)\}$. It is then sufficient to note that $\mathtt{j}$-normal forms are stable by substitutions of $\mathtt{j}$-normal forms. □

We conclude this section by showing another important property of $\lambda\mathtt{j}$ concerning the postponement of erasing steps. We first need the following lemma:

Lemma 2.11. *Let* $t \in \mathcal{T}$*. Then:*
(1) $t \to_{\mathtt{w}} \to_{\neg\mathtt{w}} t'$ *implies* $t \to_{\neg\mathtt{w}} \to_{\mathtt{w}}^+ t'$.
(2) $t \to_{\mathtt{w}}^+ \to_{\neg\mathtt{w}} t'$ *implies* $t \to_{\neg\mathtt{w}} \to_{\mathtt{w}}^+ t'$

Proof. Point 1 is by induction on the relations and case analysis. Point 2 is by induction on the length of $\to_{\mathtt{w}}^+$ using Point 1. □

Let us use $\tau : t \to^* t'$ as a notation for a reduction sequence, the symbol ';' for the concatenation of reduction sequences and $|\tau|_{\neg \mathtt{w}}$ for the number of $\to_{\neg \mathtt{w}}$ steps in τ. Then we obtain:

Lemma 2.12 ($\mathtt{w}$-postponement)**.** *Let $t \in \mathcal{T}$. If $\tau : t \to^*_{\lambda \mathtt{j}} t'$ then $\exists\ \tau' : t \to^*_{\neg \mathtt{w}} \to^*_{\mathtt{w}} t'$ s.t. $|\tau|_{\neg \mathtt{w}} = |\tau'|_{\neg \mathtt{w}}$.*

Proof. By induction on $k = |\tau|_{\neg \mathtt{w}}$. The case $k = 0$ is straightforward. Let $k > 0$. If $\tau : t \to_{\neg \mathtt{w}} u \to^*_{\lambda \mathtt{j}} t'$ then simply conclude using the i.h. on the sub-reduction $\rho : u \to^*_{\lambda \mathtt{j}} t'$. Otherwise the sequence τ starts with a $\mathtt{w}$-step. If all the steps in τ are $\mathtt{w}$, then we trivially conclude. Otherwise $\tau = \tau_{\mathtt{w}}; \to_{\neg \mathtt{w}}; \rho$ where $\tau_{\mathtt{w}}$ is the maximal prefix of τ made out of weakening steps only. By Lemma 2.11:2 we get that $t \to_{\neg \mathtt{w}} \to^+_{\mathtt{w}}; \rho\ t'$ and we conclude by applying the i.h. to $\to^+_{\mathtt{w}}; \rho$. □

2.2. **Confluence.** Confluence of calculi with ES can be easily proved by using Tait and Martin Löf's technique (see for example the case of $\lambda\mathtt{es}$ [20]). This technique is based on the definition of a simultaneous reduction relation which enjoys the diamond property. It is completely standard so we give the statements of the lemmas and omit the proofs.

The **simultaneous reduction relation** $\Rrightarrow_{\lambda \mathtt{j}}$ is defined on terms in $\mathtt{j}$-normal form as follows:

- $x \Rrightarrow_{\lambda \mathtt{j}} x$
- If $t \Rrightarrow_{\lambda \mathtt{j}} t'$, then $\lambda x.t \Rrightarrow_{\lambda \mathtt{j}} \lambda x.t'$
- If $t \Rrightarrow_{\lambda \mathtt{j}} t'$ and $u \Rrightarrow_{\lambda \mathtt{j}} u'$, then $tu \Rrightarrow_{\lambda \mathtt{j}} t'u'$
- If $t \Rrightarrow_{\lambda \mathtt{j}} t'$ and $u \Rrightarrow_{\lambda \mathtt{j}} u'$, then $(\lambda x.t)u \Rrightarrow_{\lambda \mathtt{j}} \mathtt{j}(t'[x/u'])$

Note that the third and fourth cases overlap, thus for example, $(\lambda x.II)II \Rrightarrow_{\lambda \mathtt{j}} (\lambda x.I)I$ and $(\lambda x.II)II \Rrightarrow_{\lambda \mathtt{j}} I$, where I denotes the identity function $\lambda y.y$.

A first lemma ensures that $\Rrightarrow_{\lambda \mathtt{j}}$ can be simulated by $\to_{\lambda \mathtt{j}}$.

Lemma 2.13. *If $t \Rrightarrow_{\lambda \mathtt{j}} t'$, then $t \to^*_{\lambda \mathtt{j}} t'$.*

Proof. By induction on $t \Rrightarrow_{\lambda \mathtt{j}} t'$. □

A second lemma ensures that $\to_{\lambda \mathtt{j}}$ can be projected through $\mathtt{j}(\cdot)$ on $\Rrightarrow_{\lambda \mathtt{j}}$.

Lemma 2.14. *If $t \to_{\lambda \mathtt{j}} t'$, then $\mathtt{j}(t) \Rrightarrow_{\lambda \mathtt{j}} \mathtt{j}(t')$.*

Proof. By induction on $t \to_{\lambda \mathtt{j}} t'$. □

The two lemmas combined essentially say that $\Rrightarrow_{\lambda \mathtt{j}}$ is confluent if and only if $\to^*_{\lambda \mathtt{j}}$ is confluent. Then we show the diamond property for $\Rrightarrow_{\lambda \mathtt{j}}$, which implies that $\to_{\lambda \mathtt{j}}$ is confluent:

Lemma 2.15. *The relation $\Rrightarrow_{\lambda \mathtt{j}}$ enjoys the diamond property.*

Proof. By induction on $\Rrightarrow_{\lambda \mathtt{j}}$ and case analysis. □

Then we conclude:

Theorem 2.16 (Confluence)**.** *For all $i \in \{1,2\}$, for all $t, u_i \in \mathcal{T}$ s.t. $t \to^*_{\lambda\mathtt{j}} u_i$, $\exists v$ s.t. $u_i \to^*_{\lambda\mathtt{j}} v$.*

Proof. Let $t \to^*_{\lambda\mathtt{j}} u_i$ for $i = 1, 2$. Lemma 2.14 gives $\mathtt{j}(t) \Rrightarrow^*_{\lambda\mathtt{j}} \mathtt{j}(u_i)$ for $i = 1, 2$. Lemma 2.15 implies $\Rrightarrow_{\lambda\mathtt{j}}$ is confluent so that $\exists v$ such that $\mathtt{j}(u_i) \Rrightarrow^*_{\lambda\mathtt{j}} v$ for $i = 1, 2$. We can then close the diagram with $u_i \to^*_{\mathtt{j}} \mathtt{j}(u_i) \to^*_{\lambda\mathtt{j}} v$ by Lemma 2.13. □

While confluence holds for all calculi with explicit substitutions, **metaconfluence** does not. The idea is to switch to an enriched language with a new kind of (meta)variable of the form X^{Δ}, to be intended as a named context hole expected to be replaced by terms whose free variables form a subset of Δ. This form of metaterm is for example used in the framework of higher-order unification [18]. In presence of meta-variables not all the substitutions can be computed. For instance in the metaterm $X^{y}[y/z]$ the jump $[y/z]$ is blocked. Consider:

$$(X^{\{z_1\}}Y^{\{z_2\}})[z_1/x][z_2/x] \ {}_{\mathtt{c}}\!\leftarrow (X^{\{z\}}Y^{\{z\}})[z/x] \to_{\mathtt{c}} (X^{\{z_1\}}Y^{\{z_2\}})[z_2/x][z_1/x]$$

These metaterms are different normal forms. However, it is enough to add the following equation to recover confluence:

$$t[x/u][y/v] \sim_{\mathtt{CS}} t[y/v][x/u] \text{ if } y \notin \mathtt{fv}(u) \ \&\ x \notin \mathtt{fv}(v)$$

A proof of confluence of $\lambda\mathtt{j}$ modulo CS for metaterms can be found in [38].

3. Preservation of β-Strong Normalization for $\lambda\mathtt{j}$

A reduction system $\mathcal{R}$ for a language containing the set $\mathcal{T}_\lambda$ of all λ-terms is said to enjoy the **PSN property** iff every λ-term which is β-strongly normalizing is also $\mathcal{R}$-strongly normalizing. Formally, for all $t \in \mathcal{T}_\lambda$, if $t \in \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_\mathcal{R}$.

The PSN property, when it holds, is usually non-trivial to prove. We are going to show that $\lambda\mathtt{j}$ enjoys PSN by giving a particularly compact proof. The proof technique has been developed by D. Kesner [21]; it reduces PSN to a property called **IE**, which relates termination of **I**mplicit substitution to termination of **E**xplicit substitution. It is an abstract technique not depending on the particular rules of the calculus with explicit substitutions.

A reduction system $\mathcal{R}$ for a language $\mathcal{T}_\mathcal{R}$ containing the set $\mathcal{T}_\lambda$ is said to enjoy the **IE property** iff for $n \geq 0$ and for all $t, u \in \mathcal{T}_\lambda$, $\overline{v}^1_n \subseteq \mathcal{T}_\lambda$:

$$u \in \mathcal{SN}_\mathcal{R} \ \&\ t\{x/u\}\overline{v}^1_n \in \mathcal{SN}_\mathcal{R} \ \&\ t[x/u]\overline{v}^1_n \in \mathcal{T}_\mathcal{R} \quad \text{imply} \quad t[x/u]\overline{v}^1_n \in \mathcal{SN}_\mathcal{R}$$

Of course one generally considers a system $\mathcal{R}$ which can simulate the λ-calculus, so that the following properties seem to be natural requirements to get PSN.

Theorem 3.1 (Natural Requirements for PSN)**.** *Let $\mathcal{R}$ be a calculus verifying the following facts:*

(F0) *If $\overline{t}^1_n \subseteq \mathcal{T}_\lambda \cap \mathcal{SN}_\mathcal{R}$, then $x\overline{t}^1_n \in \mathcal{SN}_\mathcal{R}$.*
(F1) *If $u \in \mathcal{T}_\lambda \cap \mathcal{SN}_\mathcal{R}$, then $\lambda x.u \in \mathcal{SN}_\mathcal{R}$.*
(F2) *If $v \in \mathcal{T}_\lambda \cap \mathcal{SN}_\mathcal{R}$ & $u\{x/v\}\overline{t}^1_n \in \mathcal{T}_\lambda \cap \mathcal{SN}_\mathcal{R}$, then $(\lambda x.u)v\overline{t}^1_n \in \mathcal{SN}_\mathcal{R}$.*
Then, $\mathcal{R}$ enjoys PSN.

Proof. We show that $t \in \mathcal{SN}_\beta$ implies $t \in \mathcal{SN}_\mathcal{R}$ by induction on the pair $\langle \eta_\beta(t), |t| \rangle$, using the lexicographic ordering. We reason by cases.

- If $t = x\overline{t}^1_n$, then $t_i \in \mathcal{SN}_\beta$ and $\langle \eta_\beta(t_i), |t_i|\rangle <_{lex} \langle \eta_\beta(t), |t|\rangle$. We have $t_i \in \mathcal{SN}_\mathcal{R}$ by the i.h. and thus $x\overline{t}^1_n \in \mathcal{SN}_\mathcal{R}$ by fact **F0**.
- If $t = \lambda x.u$, then $u \in \mathcal{SN}_\beta$ and $\langle \eta_\beta(u), |u|\rangle <_{lex} \langle \eta_\beta(t), |t|\rangle$. We have $u \in \mathcal{SN}_\mathcal{R}$ by the i.h. and thus $\lambda x.u \in \mathcal{SN}_\mathcal{R}$ by fact **F1**.
- If $t = (\lambda x.u)v\overline{t}^1_n \in \mathcal{SN}_\beta$, then $u\{x/v\}\overline{t}^1_n \in \mathcal{SN}_\beta$ and $v \in \mathcal{SN}_\beta$. Indeed, $\eta_\beta(u\{x/v\}\overline{t}^1_n) < \eta_\beta(t)$ and $\eta_\beta(v) < \eta_\beta(t)$. We have that both terms are in $\mathcal{SN}_\mathcal{R}$ by the i.h. Then **F2** guarantees that $t \in \mathcal{SN}_\mathcal{R}$. □

Now we show that $\lambda\mathtt{j}$ satisfies the three natural requirements of the last theorem, and thus it satisfies PSN.

Lemma 3.2 (Adequacy of **IE**)**.** *If $\lambda\mathtt{j}$ verifies* **IE***, then $\lambda\mathtt{j}$ satisfies PSN.*

Proof. By Theorem 3.1 it is sufficient to show **F0**, **F1** and **F2**. The first two properties are straightforward. For the third one, assume $v \in \mathcal{T}_\lambda \cap \mathcal{SN}_{\lambda\mathtt{j}}$ and $u\{x/v\}\overline{t}^1_n \in \mathcal{T}_\lambda \cap \mathcal{SN}_{\lambda\mathtt{j}}$. Then in particular $u, v, \overline{t}^1_n \in \mathcal{T}_\lambda \cap \mathcal{SN}_{\lambda\mathtt{j}}$. We show that $t = (\lambda x.u)v\overline{t}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}}$ by induction on $\eta_{\lambda\mathtt{j}}(u) + \eta_{\lambda\mathtt{j}}(v) + \Sigma_i\, \eta_{\lambda\mathtt{j}}(t_i)$. For that, it is sufficient to show that every $\lambda\mathtt{j}$-reduct of t is in $\mathcal{SN}_{\lambda\mathtt{j}}$. If the $\lambda\mathtt{j}$-reduct of t is internal we conclude by the i.h. Otherwise $t = u[x/v]\overline{t}^1_n$ which is in $\mathcal{SN}_{\lambda\mathtt{j}}$ by the **IE** property. □

As a consequence, in order to get PSN for $\lambda\mathtt{j}$ we only need to prove the **IE** property. For that, we first generalise the **IE** property in order to deal with possibly many substitutions.

A reduction system $\mathcal{R}$ for a language $\mathcal{T}_\mathcal{R}$ containing the set $\mathcal{T}_\lambda$ is said to enjoy the **Generalised IE property**, written **GIE,** iff for all $t, \overline{u}^1_m\ (m \geq 1), \overline{v}^1_n\ (n \geq 0)$ in $\mathcal{T}_\mathcal{R}$, if $\overline{u}^1_m \subseteq \mathcal{SN}_\mathcal{R}$ & $t\{x_i/u_i\}^1_m\overline{v}^1_n \in \mathcal{SN}_\mathcal{R}$, then $t[x_i/u_i]^1_m\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}}$, where $x_i \neq x_j$ for $i, j = 1 \dots m$ and $x_i \notin \mathtt{fv}(u_j)$ for $i, j = 1 \dots m$.

Theorem 3.3 (**GIE** for $\lambda\mathtt{j}$)**.** *The $\lambda\mathtt{j}$-calculus enjoys the* **GIE** *property.*

Notation: To improve readability of the proof we shall abbreviate the notation $[x_i/u_i]^1_m$ by $[\cdot]^1_m$. Similarly for implicit substitutions.

Proof. Suppose $\overline{u}^1_m \in \mathcal{SN}_{\lambda\mathtt{j}}$ & $t\{x_i/u_i\}^1_m\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}}$. We show $t_0 = t[x_i/u_i]^1_m\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}}$ by induction on:

$$\langle \eta_{\lambda\mathtt{j}}(t\{x_i/u_i\}^1_m\overline{v}^1_n),\quad \mathtt{v}_{\overline{x}^1_m}(t),\quad \eta_{\lambda\mathtt{j}}(\overline{u}^1_m)\rangle$$

where $\mathtt{v}_{x_i}(t) = 3^{|t|_{x_i}}$ and $\mathtt{v}_{\overline{x}^1_m}(t) = \Sigma_{i\in m}\mathtt{v}_{x_i}(t)$.

To show $t_0 \in \mathcal{SN}_{\lambda\mathtt{j}}$ it is sufficient to show that every $\lambda\mathtt{j}$-reduct of t_0 is in $\mathcal{SN}_{\lambda\mathtt{j}}$.

- $t_0 \to_{\lambda\mathtt{j}} t[\cdot]^1_{j-1}[x_j/u'_j][\cdot]^{j+1}_m\overline{v}^1_n = t'_0$ with $u_j \to_{\lambda\mathtt{j}} u'_j$. Then we get:
  - $\eta_{\lambda\mathtt{j}}(t\{\cdot\}^1_{j-1}\{x_j/u'_j\}\{\cdot\}^{j+1}_m\ \overline{v}^1_n) \leq \eta_{\lambda\mathtt{j}}(t\{\cdot\}^1_m\overline{v}^1_n)$,
  - $\mathtt{v}_{\overline{x}^1_m}$ does not change, and
  - $\eta_{\lambda\mathtt{j}}(\overline{u}^1_{j-1}u'_j\overline{u}^{j+1}_m) < \eta_{\lambda\mathtt{j}}(\overline{u}^1_m)$.

  We conclude by the i.h. since $\overline{u}^1_{j-1}u'_j\overline{u}^{j+1}_m \in \mathcal{SN}_{\lambda\mathtt{j}}$ and our hypothesis $t\{x_i/u_i\}^1_m\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}}$ is equal or reduces to $t\{\cdot\}^1_{j-1}\{x_j/u'_j\}\{\cdot\}^{j+1}_m\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}}$ (depending on $|t|_{x_j}$).
- $t_0 \to_{\lambda\mathtt{j}} t'[\cdot]^1_m\overline{v}^1_n = t'_0$ with $t \to_{\lambda\mathtt{j}} t'$. Then we have that:

  $$\eta_{\lambda\mathtt{j}}(t'\{\cdot\}^1_m\overline{v}^1_n) < \eta_{\lambda\mathtt{j}}(t\{\cdot\}^1_m\overline{v}^1_n)$$

  We conclude by the i.h. since $t'\{\cdot\}^1_m\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}}$.

- $t_0 \to_{\lambda \mathtt{j}} t[\cdot]^1_m v_1 \dots v_i' \dots v_n = t_0'$ with $v_i \to_{\lambda \mathtt{j}} v_i'$. Then we have that:
$$\eta_{\lambda \mathtt{j}}(t\{\cdot\}^1_m v_1 \dots v_i' \dots v_n) < \eta_{\lambda \mathtt{j}}(t\{\cdot\}^1_m \overline{v}^1_n)$$
We conclude by the i.h. since $t\{\cdot\}^1_m v_1 \dots v_i' \dots v_n \in \mathcal{SN}_{\lambda \mathtt{j}}$.
- $t_0 \to_{\mathtt{w}} t[\cdot]^1_{j-1}[\cdot]^{j+1}_m \overline{v}^1_n = t_0'$, with $|t|_{x_j} = 0$. Then we have that:
$$\eta_{\lambda \mathtt{j}}(t\{\cdot\}^1_{j-1}\{\cdot\}^{j+1}_m \overline{v}^1_n) = \eta_{\lambda \mathtt{j}}(t\{\cdot\}^1_m \overline{v}^1_n)$$
But $\mathtt{v}_{\overline{x}^1_{j-1}\overline{x}^{j+1}_m}(t) < \mathtt{v}_{\overline{x}^1_m}(t)$. We conclude by the i.h. since $t\{\cdot\}^1_{j-1}\{\cdot\}^{j+1}_m \overline{v}^1_n = t\{\cdot\}^1_m \overline{v}^1_n \in \mathcal{SN}_{\lambda \mathtt{j}}$ by hypothesis.
- $t_0 \to_{\mathtt{d}} t[\cdot]^1_{j-1}\{x_j/u_j\}[\cdot]^{j+1}_m \overline{v}^1_n = t_0'$ with $|t|_{x_j} = 1$. Note that $t_0' = t\{x_j/u_j\}[\cdot]^1_{j-1}[\cdot]^{j+1}_m \overline{v}^1_n$. Then we get:
$$\eta_{\lambda \mathtt{j}}(t\{x_j/u_j\}\{\cdot\}^1_{j-1}\{\cdot\}^{j+1}_m \overline{v}^1_n) = \eta_{\lambda \mathtt{j}}(t\{\cdot\}^1_m \overline{v}^1_n)$$
Since the jumps are independent, then $(\overline{x}^1_{j-1} \cup \overline{x}^{j+1}_m) \cap \mathtt{fv}(u_j) = \emptyset$ implies $\mathtt{v}_{\overline{x}^1_{j-1}\overline{x}^{j+1}_m}(t\{x_j/u_j\}) < \mathtt{v}_{\overline{x}^1_m}(t)$. We conclude since $t\{\cdot\}^1_{j-1}\{x_j/u_j\}\{\cdot\}^{j+1}_m \overline{v}^1_n = t\{\cdot\}^1_m \overline{v}^1_n \in \mathcal{SN}_{\lambda \mathtt{j}}$ by hypothesis.
- $t_0 \to_{\mathtt{c}} t_{[y]_{x_j}}[\cdot]^1_{j-1}[x_j/u_j][y/u_j][\cdot]^{j+1}_m \overline{v}^1_n = t_0'$ with $|t|_{x_j} \geq 2$ and y fresh. Then,
$\eta_{\lambda \mathtt{j}}(t_{[y]_{x_j}}\{\cdot\}^1_{j-1}\{x_j/u_j\}\{y/u_j\}\{\cdot\}^{j+1}_m \overline{v}^1_n) = \eta_{\lambda \mathtt{j}}(t\{\cdot\}^1_m \overline{v}^1_n))$ and
$\mathtt{v}_{\overline{x}^1_{j-1}x_j y \overline{x}^{j+1}_m}(t_{[y]_{x_j}}) < \mathtt{v}_{\overline{x}^1_m}(t)$. In order to apply the i.h. to $t_{[y]_{x_j}}$ we need:
  - $\overline{u}^1_{j-1}, u_j, u_j, \overline{u}^{j+1}_m \in \mathcal{SN}_{\lambda \mathtt{j}}$. This holds by hypothesis.
  - $t_{[y]_{x_1}}\{\cdot\}^1_{j-1}\{x_j/u_j\}\{y/u_j\}\{\cdot\}^{j+1}_m \overline{v}^1_n \in \mathcal{SN}_{\lambda \mathtt{j}}$. This holds since the term is equal to $t\{\cdot\}^1_m \overline{v}^1_n$ which is $\mathcal{SN}_{\lambda \mathtt{j}}$ by hypothesis.

  Note that this is the case that forces the use of a generalised sequence of substitutions: if we were proving the statement for $t[x/u]\overline{v}^1_n$ using as hypothesis $u \in \mathcal{SN}_{\lambda \mathtt{j}}$ & $t\{x/u\}\overline{v}^1_n \in \mathcal{SN}_{\lambda \mathtt{j}}$ then there would be no way to use the i.h. to get $t_{[y]_x}[x/u][y/u]\overline{v}^1_n \in \mathcal{SN}_{\lambda \mathtt{j}}$.
- $t_0 = (\lambda x.t')[\cdot]^1_m v_1 \overline{v}^2_n \to_{\mathtt{dB}} t'[x/v_1][\cdot]^1_m \overline{v}^2_n = t_0'$. We have that:
$$u_0 = (\lambda x.t')\{\cdot\}^1_m v_1 \overline{v}^2_n \in \mathcal{SN}_{\lambda \mathtt{j}}$$
holds by hypothesis. Then:
$$u_0 \to_{\mathtt{dB}} t'\{\cdot\}^1_m[x/v_1]\overline{v}^2_n = t'[x/v_1]\{\cdot\}^1_m \overline{v}^2_n = u_0'$$
Thus $\eta_{\lambda \mathtt{j}}(u_0') < \eta_{\lambda \mathtt{j}}(u_0)$ and $u_0' \in \mathcal{SN}_{\lambda \mathtt{j}}$. Since $\overline{u}^1_m \in \mathcal{SN}_{\lambda \mathtt{j}}$ by hypothesis we can apply the i.h. and get $t_0' \in \mathcal{SN}_{\lambda \mathtt{j}}$. □

The following is a consequence of Thereom 3.3: just take the number of substitutions m to be 1 and consider only the **GIE** property for $\mathcal{T}_\lambda \subset \mathcal{T}$.

Corollary 3.4 (**IE** for $\lambda \mathtt{j}$)**.** *The $\lambda \mathtt{j}$-calculus enjoys the* **IE** *property.*

Corollary 3.4, then Lemma 3.2 and finally Theorem 3.1 imply:

Corollary 3.5 (PSN for $\lambda \mathtt{j}$)**.** *The $\lambda \mathtt{j}$-calculus enjoys PSN, i.e. if $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_{\lambda \mathtt{j}}$.*

Note that Lemma 3.2 and Theorem 3.3, which contains the arguments for PSN, do not use full composition, nor termination of $\to_{\mathtt{j}}$, confluence or postponement of erasures: none of the properties of $\lambda \mathtt{j}$ plays a role in this compact proof of PSN, which is quite surprising. The crucial point is the formulation at a distance of the rewriting rules. Indeed, we will

later show that such a simple proof does not longer work when rules propagating jumps are added to the system.

4. An equational theory for $\lambda\mathtt{j}$

Sections 2 and 3 show that the basic theory of $\lambda\mathtt{j}$ enjoys good properties such as full composition, confluence and PSN. In most calculi with explicit substitutions, where substitutions are propagated through constructors and do not act at a distance, full composition can only be obtained by adding an equivalence relation $\equiv_{\mathtt{CS}}$ defined as the contextual and reflexive-transitive closure of the following equation:

$$t[x/s][y/v] \quad \sim_{\mathtt{CS}} \quad t[y/v][x/s] \qquad \text{if } x \notin \mathtt{fv}(v) \ \&\ y \notin \mathtt{fv}(s)$$

Otherwise a term like $x[y/z][x/w]$ cannot reduce to its implicit form $w[y/z] = x[y/z]\{x/w\}$ (and so full composition does not hold). Interestingly, $\lambda\mathtt{j}$ enjoys full composition without using equation $\mathtt{CS}$, which is remarkable since plain rewriting is much easier than rewriting modulo an equivalence relation.

However, as mentioned at the end of Section 2.2, the equation $\mathtt{CS}$ is necessary to recover confluence on metaterms. It is then natural to wonder what happens when $\equiv_{\mathtt{CS}}$ is added to $\lambda\mathtt{j}$. The answer is extremely positive since $\equiv_{\mathtt{CS}}$ preserves all the good properties of $\lambda\mathtt{j}$, and this holds in a very strong sense. In fact, $\equiv_{\mathtt{CS}}$ is a strong bisimulation for $(\mathcal{T}, \to_{\lambda\mathtt{j}})$ (*cf.* Lemma 4.2), so that $\equiv_{\mathtt{CS}}$ can be postponed w.r.t. $\to_{\lambda\mathtt{j}}$ (*cf.* Lemma 1.2) and $\lambda\mathtt{j}$ modulo $\equiv_{\mathtt{CS}}$ enjoys PSN (*cf.* Lemma 1.1:2).

As already mentioned in the introduction, $\lambda\mathtt{j}$-terms and $\lambda\mathtt{j}$-dags [1] are strongly bisimilar, but the translation of $\lambda\mathtt{j}$-terms to $\lambda\mathtt{j}$-dags is not injective, *i.e.* there are different $\lambda\mathtt{j}$-terms which are mapped to the *same* $\lambda\mathtt{j}$-dag. It is then interesting to characterise the quotient induced by the translation [1], which turns out to be $\equiv_{\mathtt{CS}}$: indeed $t \equiv_{\mathtt{CS}} u$ if and only if t and u are mapped to the same $\lambda\mathtt{j}$-dag G, and since they both behave like G (*i.e.* are strongly bisimilar to G), then they behave the same (*i.e.* they are strongly bisimilar).

The $\lambda\mathtt{j}$-calculus is also interesting since it can be mapped to another graphical language, Danos' and Regnier's **Pure Proof-Nets**, being able to capture *untyped* λ-calculus. It is possible to endow Pure Proof-Nets with an operational semantics[1] which makes them strongly bisimilar to $\lambda\mathtt{j}$. The quotient induced by the translation from $\lambda\mathtt{j}$-terms into Pure Proof-Nets is given by the **graphical equivalence** $\equiv_{\mathtt{o}}$ which is the contextual and reflexive-transitive closure of the equations in Figure 2.

$$\begin{array}{llll} t[x/s][y/v] & \sim_{\mathtt{CS}} & t[y/v][x/s] & \text{if } x \notin \mathtt{fv}(v) \ \&\ y \notin \mathtt{fv}(s) \\ \lambda y.t[x/s] & \sim_{\sigma_1} & (\lambda y.t)[x/s] & \text{if } y \notin \mathtt{fv}(s) \\ t[x/s]v & \sim_{\sigma_2} & (tv)[x/s] & \text{if } x \notin \mathtt{fv}(v) \end{array}$$

Figure 2: The graphical equivalence $\equiv_{\mathtt{o}}$

[1] Danos' and Regnier's original operational semantics does not match exactly $\lambda\mathtt{j}$ because they use a big-steps rule for eliminating exponential cuts, which corresponds to use just one substitution rule $t[x/u] \to t\{x/u\}$. However, the refinement of Pure Proof-Nets where duplications are done small-steps is very natural from an explicit substitution point of view, altough — to our knowledge — it has never been considered before.

This means that Pure Proof-Nets quotient more than $\lambda\mathtt{j}$-dags[2]. As for $\equiv_{\mathtt{CS}}$, $\equiv_{\mathtt{o}}$ is a strong bisimulation (*cf.* Lemma 4.2), and thus confluence and PSN of $\lambda\mathtt{j}$ automatically lift to $\to_{\lambda\mathtt{j}/\mathtt{o}}$ (*cf.* Theorem 4.3), which is the reduction relation $\lambda\mathtt{j}$ modulo $\equiv_{\mathtt{o}}$.

Another way to explain the $\mathtt{o}$-equivalence is by means of *linear* constructors. Indeed, the body of an abstraction cannot be duplicated nor erased by the abstraction itself—in this sense an abstraction is *linear* in its body. Similarly, explicit substitutions are linear with respect to their left subterm, while they are *non-linear* with respect to their right subterm, *i.e.* the content of the jump, which may be duplicated or discarded. Applications are linear in their left subterm but they are non-linear in their argument, because they can wrap it in a jump. This linear/non-linear classification reflects the fact that jumps and arguments (and only them) are associated to !-boxes in Proof-Nets, the non-linear construction of Linear Logic. The equations defining $\equiv_{\mathtt{o}}$ can be understood as a permutation between a jump and a linear subterm of the adjacent constructor.

It is then natural to wonder if $\equiv_{\mathtt{o}}$ can be extended with equations permuting jumps with non-linear subterms (see Figure 4, page 22), without breaking confluence and PSN. The answer is yes; the obtained equational theory is called the **substitution equivalence** $\equiv_{\mathtt{obox}}$, and the fact that $\lambda\mathtt{j}$ modulo $\equiv_{\mathtt{obox}}$ enjoys PSN is the main result of this paper.

Extending $\equiv_{\mathtt{o}}$ to non-linear permutations is delicate from a termination point of view, since the use of non-linear equations affects reduction lengths. Indeed, the natural but naïve extension of $\equiv_{\mathtt{o}}$ breaks PSN. By analyzing a counter-example to PSN we define $\equiv_{\mathtt{obox}}$ so that PSN turns out to be true. The proof of this fact, however, is more involved than that for $\equiv_{\mathtt{CS}}$ and $\equiv_{\mathtt{o}}$, mainly because $\equiv_{\mathtt{obox}}$ is not a strong bisimulation. Therefore, we shall develop a new technique for proving PSN modulo $\equiv_{\mathtt{obox}}$.

Section 4.1 starts over by explaining the equivalence $\equiv_{\mathtt{o}}$ in terms of Regnier's σ-equivalence [37], providing a different point of view with respect to what was already mentioned. Section 4.2 discusses how to extend $\equiv_{\mathtt{o}}$ to $\equiv_{\mathtt{obox}}$ by showing the difficulties to prove PSN for the obtained extension. Section 5 develops the proof of PSN for $\lambda\mathtt{j}$ modulo $\equiv_{\mathtt{obox}}$.

4.1. **The graphical equivalence. Regnier's equivalence** $\equiv_{\hat{\sigma}}$ is the smallest equivalence on λ-terms closed by contexts and containing the equations in Figure 3.

$$\begin{array}{llll} (\lambda x.\lambda y.t)u & \sim_{\hat{\sigma}_1} & \lambda y.((\lambda x.t)u) & \text{if } y \notin \mathtt{fv}(u) \\ (\lambda x.tv)u & \sim_{\hat{\sigma}_2} & (\lambda x.t)uv & \text{if } x \notin \mathtt{fv}(v) \end{array}$$

Figure 3: The equivalence $\equiv_{\hat{\sigma}}$

Regnier proved that two $\hat{\sigma}$-equivalent terms have essentially the same *operational* behavior: $\equiv_{\hat{\sigma}}$ is contained in the equational theory generated by β-reduction, *i.e.* $\equiv_{\hat{\sigma}} \subset \equiv_{\beta}$, and if $t \equiv_{\hat{\sigma}} t'$ then the maximal β-reduction sequences from t and t' have the same length (the so-called *Barendregt's norm*). That is why Regnier calls $\equiv_{\hat{\sigma}}$ an *operational equivalence*.

[2] $\lambda\mathtt{j}$-dags can be mapped on Pure Proof-Nets, and once again the map is a strong bisimulation.

It is then natural to expect that the previous property can be *locally* reformulated in terms of a strong bisimulation, namely,

$$\begin{array}{ccc} t & \to_\beta & u \\ \equiv_{\hat\sigma} & & \\ t' & & \end{array} \qquad \text{implies} \qquad \begin{array}{ccc} t & \to_\beta & u \\ \equiv_{\hat\sigma} & & \equiv_{\hat\sigma} \\ t' & \to_\beta & u' \end{array}$$

Unfortunately, this is not the case. Consider the following example, where grey boxes are used to help the identification of redexes and their reductions:

$$\begin{array}{ccccccc} t & = & \lambda y.(\,(\lambda x.y)z_1\,)z_2 & \to_\beta & (\lambda x.z_2)z_1 & = & u \\ & & \equiv_{\hat\sigma_1} & & \not\equiv_{\hat\sigma} & & \\ t' & = & (\,(\lambda x.(\lambda y.y))z_1\,)z_2 & \to_\beta & (\lambda y.y)z_2 & = & u' \end{array}$$

The term t' has only one redex whose reduction gives u' which is not $\equiv_{\hat\sigma}$-equivalent to u, the reduct of t. The diagram can be completed only by unfolding the whole reduction:

$$\begin{array}{ccccccc} t & = & \lambda y.(\,(\lambda x.y)z_1\,)z_2 & \to_\beta & (\lambda x.z_2)z_1 & \to_\beta & z_2 \\ & & \equiv_{\hat\sigma_1} & & & & = (\subseteq \equiv_{\hat\sigma}) \\ t' & = & (\,(\lambda x.(\lambda y.y))z_1\,)z_2 & \to_\beta & (\lambda y.y)z_2 & \to_\beta & z_2 \end{array}$$

Note that the second step from t' reduces a *created* redex.

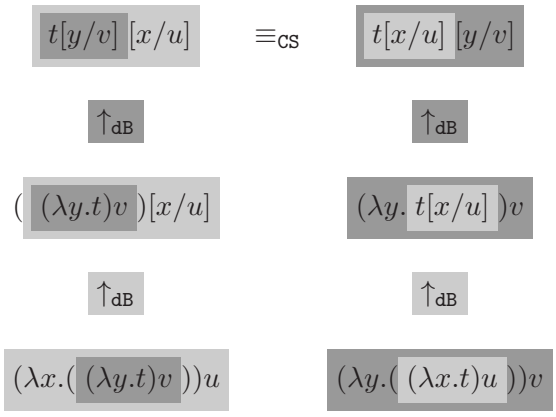
We are now going to analyze $\equiv_{\hat\sigma}$ in the framework of $\lambda\mathtt{j}$. For that, Regnier's equivalence can be understood on $\lambda\mathtt{j}$-terms by first removing the $\mathtt{dB}$-redexes. Indeed, let us take the clauses defining $\equiv_{\hat\sigma}$ and let us make a $\mathtt{dB}$-reduction step on both sides, thus eliminating the multiplicative redexes as in Regnier's definition:

$$\begin{array}{ccccccc} (\lambda x.\lambda y.t)u & \sim_{\hat\sigma_1} & \lambda y.(\,(\lambda x.t)u\,) & & (\,(\lambda x.t)u\,)v & \sim_{\hat\sigma_2} & (\lambda x.(tv))u \\ \downarrow_{\mathtt{dB}} & & \downarrow_{\mathtt{dB}} & & \downarrow_{\mathtt{dB}} & & \downarrow_{\mathtt{dB}} \\ (\lambda y.t)[x/u] & & \lambda y.(\,t[x/u]\,) & & t[x/u]\,v & & (tv)[x/u] \end{array}$$

Now, $\equiv_{\hat\sigma}$ can be seen as a change of the positions of jumps in a given term and particularly as a permutation equivalence of jumps concerning the linear constructors of the calculus.

This is not so surprising since such permutations turn into simple equalities when one extends the standard translation of λ-calculus into Linear Logic Proof-Nets to $\lambda\mathtt{j}$-terms (see for example [24]). Another interesting observation is the relationship between $\equiv_{\hat\sigma}$ and the equivalence $\equiv_{\mathtt{CS}}$ introduced in Section 2.2. To understand this point we proceed the other

way around by expanding jumps into β-redexes:



Note that the relation between the resulting terms is contained in $\equiv_{\hat{\sigma}}$, that is why it was not visible in λ-calculus:

$$(\lambda x.((\lambda y.t)v))u \quad \sim_{\hat{\sigma}_2} \quad (\lambda x.\lambda y.t)uv \quad \sim_{\hat{\sigma}_1} \quad (\lambda y.((\lambda x.t)u))v$$

In [1] it has been proved that two $\lambda\mathtt{j}$-terms t and t' are translated to the same Pure Proof-Net if and only if $t \equiv_{\sigma,\mathtt{CS}} t'$. More precisely, this relation can be given by the **graphical equivalence** $\equiv_{\mathtt{o}}$ already defined in Figure 2.

The equations defining $\equiv_{\mathtt{o}}$ are specified by means of *local* permutations, but it is also possible to define $\equiv_{\mathtt{o}}$ in terms of global permutations. First, define a **spine context** S as:

$$S ::= \Box \mid \lambda x.S \mid St \mid S[x/t]$$

and then define $\equiv_{\mathtt{o}}$ as the context closure of the following equation $\sim_{\mathtt{o}}$:

$$S[\![t[x/u]]\!] \sim_{\mathtt{o}} S[\![t]\!][x/u] \qquad \text{if } \mathtt{bs}(S) \cap \mathtt{fv}(u) = \emptyset \text{ and } |t|_x = |S[\![t]\!]|_x$$

The two definitions are easily seen to be equivalent. We shall now prove that $\equiv_{\mathtt{o}}$ is a strong bisimulation, which will immediately imply (Lemma 1.1) that $\equiv_{\mathtt{o}}$ preserves reduction lengths. This property is stronger than the one proved by Regnier for $\equiv_{\hat{\sigma}}$, since it holds for any reduction sequence, not only for the maximal ones.

Lemma 4.1. *Let* $\mathtt{E}$ *be the equivalence relation* $\mathtt{CS}$ *or* $\mathtt{o}$*, and* $t, t' \in \mathcal{T}$ *s.t.* $t \equiv_{\mathtt{E}} t'$*. Let* $u \in \mathcal{T}$*. Then:*

(1) $|t|_x = |t'|_x$.
(2) *For all* $S \subseteq \mathtt{pos}_x(t)$ *there is* $S' \subseteq \mathtt{pos}_x(t')$ *s.t.* $|S| = |S'|$ *and* $R_y^{S,x}(t) \equiv_{\mathtt{E}} R_y^{S',x}(t')$.
(3) $t\{x/u\} \equiv_{\mathtt{E}} t'\{x/u\}$.
(4) $u\{x/t\} \equiv_{\mathtt{E}} u\{x/t'\}$.

Proof. Straightforward inductions. □

Lemma 4.2. *The relations* $\equiv_{\mathtt{CS}}$ *and* $\equiv_{\mathtt{o}}$ *are strong bisimulations for* $\lambda\mathtt{j}$.

Proof. We prove the statement for $\equiv_{\mathtt{o}}$. The proof for $\equiv_{\mathtt{CS}}$ is obtained by simply forgetting the cases $\{\sim_{\sigma_1}, \sim_{\sigma_2}\}$. Assume $t_0 \equiv_{\mathtt{o}} t_1$ holds in n-steps, which is written as $t_0 \equiv_{\mathtt{o}}^n t_1$, and let $t_1 \to_{\lambda\mathtt{j}} s_1$. We show $\exists\ s_0$ s.t. $t_0 \to_{\lambda\mathtt{j}} s_0 \equiv_{\mathtt{o}} s_1$ by induction on n.

The inductive step for $n > 1$ is straightforward. For $n = 1$ we reason by induction on the definition of $t_0 \equiv_{\mathtt{o}}^1 t_1$, given by the closure under contexts of the equations $\{\sim_{\mathtt{CS}}, \sim_{\sigma_1}, \sim_{\sigma_2}\}$.

We only show here the cases where $t_0 \equiv_{\mathtt{o}} t_1$ is contextual, all the other ones being straightforward.

- If $t_0 = t[x/u] \equiv_{\mathtt{o}} t'[x/u] = t_1 \to_{\lambda\mathtt{j}} t'[x/u'] = s_1$, where $t \equiv_{\mathtt{o}} t'$ and $u \to_{\lambda\mathtt{j}} u'$, then we close the diagram by $t_0 \to_{\lambda\mathtt{j}} t[x/u'] \equiv_{\mathtt{o}} s_1$.
- The case $t_0 = t[x/u] \equiv_{\mathtt{o}} t[x/u'] = t_1 \to_{\lambda\mathtt{j}} t'[x/u'] = s_1$, where $u \equiv_{\mathtt{o}} u'$ and $t \to_{\lambda\mathtt{j}} t'$ is analogous to the previous one.
- If $t_0 = t[x/u] \equiv_{\mathtt{o}} t'[x/u] = t_1 \to_{\lambda\mathtt{j}} t''[x/u] = s_1$, where $t \equiv_{\mathtt{o}} t' \to_{\lambda\mathtt{j}} t''$, then by the i.h. $\exists\, t'''$ s.t. $t \to_{\lambda\mathtt{j}} t''' \equiv_{\mathtt{o}} t''$. We close the diagram by $t_0 \to_{\lambda\mathtt{j}} t'''[x/u] \equiv^*_{\mathtt{o}} s_1$.
- The case $t_0 = t[x/u] \equiv_{\mathtt{o}} t[x/u'] = t_1 \to_{\lambda\mathtt{j}} t[x/u''] = s_1$, where $u \equiv_{\mathtt{o}} u' \to_{\lambda\mathtt{j}} u''$ is analogous to the previous one.
- If $t_0 = t[x/u] \equiv_{\mathtt{o}} t[x/u'] = t_1 \to_{\mathtt{w}} t = s_1$, where $u \equiv_{\mathtt{o}} u'$ and $|t|_x = 0$, then $t_0 \to_{\mathtt{w}} t = s_1$.
- If $t_0 = t[x/u] \equiv_{\mathtt{o}} t'[x/u] = t_1 \to_{\mathtt{w}} t = s_1$, where $t \equiv_{\mathtt{o}} t'$ and $|t'|_x = 0$, then the previous remark implies $|t|_x = 0$ and we close the diagram by $t_0 \to_{\mathtt{w}} t \equiv_{\mathtt{o}} t' = s_1$.
- If $t_0 = t[x/u] \equiv_{\mathtt{o}} t[x/u'] = t_1 \to_{\mathtt{c}} t_{[y]_x}[x/u'][y/u'] = s_1$, where $u \equiv_{\mathtt{o}} u'$ and $|t|_x > 1$, then we close the diagram by $t_0 \to_{\mathtt{c}} t = t_{[y]_x}[x/u][y/u] \equiv^2_{\mathtt{o}} t_{[y]_x}[x/u'][y/u']$.
- If $t_0 = t[x/u] \equiv_{\mathtt{o}} t'[x/u] = t_1 \to_{\mathtt{c}} t'_{[y]_x}[x/u][y/u] = s_1$, where $t \equiv_{\mathtt{o}} t'$ and $|t'|_x > 1$, then we first write $t'_{[y]_x}$ as $R_y^{S',x}(t')$, where $S' \subset \mathtt{pos}_{t'}(x)$ and $|S'| \geq 2$. Lemma 4.1:1 gives $|t|_x > 1$ and Lemma 4.1:2 gives $S \subset \mathtt{pos}_t(x)$ verifying $|S| = |S'|$ and $R_y^{S',x}(t') \equiv_{\mathtt{o}} R_y^{S,x}(t)$. Then, we close the diagram with $t_0 \to_{\mathtt{c}} R_y^{S,x}(t)[x/u][y/u] \equiv_{\mathtt{o}} t'_{[y]_x}[x/u][y/u]$.
- If $t_0 = t[x/u] \equiv_{\mathtt{o}} t[x/u'] = t_1 \to_{\mathtt{d}} t\{x/u'\} = s_1$, where $u \equiv_{\mathtt{o}} u'$ and $|t|_x = 1$, then $t[x/u] \to_{\mathtt{d}} t\{x/u\} \equiv_{\mathtt{o}} t\{x/u'\}$, where the last equivalence holds by Lemma 4.1:4.
- If $t_0 = t[x/u] \equiv_{\mathtt{o}} t'[x/u] = t_1 \to_{\mathtt{d}} t'\{x/u\} = s_1$, where $t \equiv_{\mathtt{o}} t'$ and $|t'|_x = 1$. Then, $t[x/u] \to_{\mathtt{d}} t\{x/u\} \equiv_{\mathtt{o}} t'\{x/u\}$ where the last equivalence holds by Lemma 4.1:1-3. □

A consequence (*cf.* Lemma 1.2) of the previous lemma is that both $\equiv_{\mathtt{CS}}$ and $\equiv_{\mathtt{o}}$ can be postponed, which implies in particular the following.

Theorem 4.3. *The reduction systems* $(\mathcal{T}, \to_{\lambda\mathtt{j}/\mathtt{CS}})$ *and* $(\mathcal{T}, \to_{\lambda\mathtt{j}/\mathtt{o}})$ *are both confluent and enjoy PSN.*

Proof. Confluence follows from Lemma 4.2 and Theorem 2.16 by application of Lemma 1.2:2, while PSN follows from Lemma 4.2 and Corollary 3.5 by application of Lemma 1.1. □

Actually, $\to_{\lambda\mathtt{j}/\mathtt{o}}$ is equal to $\equiv_{\mathtt{o}} \to_{\lambda\mathtt{j}} \equiv_{\mathtt{o}}$. In the framework of rewriting modulo an equivalence relation there are various, non-equivalent, forms of confluence. The one given by Theorem 4.3 is the weakest one, but the *Church-Rosser modulo* property also holds in our framework.

Theorem 4.4 (Church-Rosser modulo CS and o)**.** *Let* E *be the equivalence relation* CS *or* o*. If* $t_0 \equiv_{\mathtt{E}} t_1$*,* $t_0 \to^*_{\lambda\mathtt{j}/\mathtt{E}} u_0$ *and* $t_1 \to^*_{\lambda\mathtt{j}/\mathtt{E}} u_1$*, then* $\exists v_0, v_1$ *s.t.* $u_0 \to^*_{\lambda\mathtt{j}/\mathtt{E}} v_0$*,* $u_1 \to^*_{\lambda\mathtt{j}/\mathtt{E}} v_1$ *and* $v_0 \equiv_{\mathtt{E}} v_1$*.*

Proof. By Lemma 4.2 and Lemma 1.2. □

We finish this section with the following interesting property.

Lemma 4.5. *The reduction relation* j/o *is strongly normalizing.*

Proof. The proof uses the measure jm() used to prove Lemma 2.9 and the fact that $t \equiv_{\mathtt{o}} t'$ implies $\mathtt{jm}(t) = \mathtt{jm}(t')$. □

4.2. **The substitution equivalence.** Composition of explicit substitutions is a sensible topic in the literature, it is interesting to know if $\lambda\mathtt{j}$ can be extended with a safe notion of (structural) composition.

The structural λ-calculus is peculiar since composition of substitution is provided natively, but only *implicitly* and at a distance. Indeed, a term $t[x/u][y/v]$ s.t. $y \in \mathtt{fv}(u)$ & $y \in \mathtt{fv}(t)$ reduces in various steps to:

$$t[x/u\{y/v\}][y/v]$$

but not to the *explicit composition* $t[x/u[y/v]][y/v]$. One of the aims of this paper is to prove that adding explicit composition to $\lambda\mathtt{j}$ preserves PSN and confluence.

The second aim concerns *explicit decomposition*. Indeed, some calculi [36, 31, 40, 16, 15] explicitly *decompose* substitutions, *i.e.* reduce $t[x/u[y/v]]$ to $t[x/u][y/v]$. We show that PSN and confluence hold even when extending $\lambda\mathtt{j}$ with such a rule.

More generally, having a core system, $\lambda\mathtt{j}$, whose operational semantics does not depend on propagations, we study how to modularly add propagations by keeping the good properties. We have already shown that $\lambda\mathtt{j}$ is stable with respect to the graphical equivalence, which can be seen as handling propagations of jumps with respect to *linear* constructors. We proved that $\lambda\mathtt{j/o}$ is confluent and enjoys PSN (Theorem 4.3). What we investigate here is if we can extend it to propagations with respect to *non-linear* constructors.

The idea is to extend $\equiv_{\mathtt{o}}$ to $\equiv_{\mathtt{n}}$, where $\equiv_{\mathtt{n}}$ is the the contextual and reflexive-transitive closure of the relation generated by $\{\mathtt{CS}, \sigma_1, \sigma_2\}$ plus:

$$\begin{array}{llll} (tv)[x/u] & \sim_{\mathtt{box}_1^0} & tv[x/u] & \text{if } x \notin \mathtt{fv}(t) \\ t[y/v][x/u] & \sim_{\mathtt{box}_2^0} & t[y/v[x/u]] & \text{if } x \notin \mathtt{fv}(t) \end{array}$$

In terms of global permutations $\equiv_{\mathtt{n}}$ can be defined as the context closure of $C[\![t[x/u]]\!] \sim_{\mathtt{n}} C[\![t]\!][x/u]$ where $|t|_x = |C[\![t]\!]|_x$, and C is *any* context (not just a spine context) which does not capture the variables of u. These equations are constructor preserving (same kind and number of constructors), in contrast to more traditional explicit substitution calculi containing for instance the following rule:

$$(tu)[y/v] \quad \to_{@} \quad t[y/v]u[y/v]$$

which achieves two actions at the same time: duplication and propagation of a jump. In $\lambda\mathtt{j/n}$ there is a neat separation between propagations and duplications, so that no propagation affects the number of constructors. The rule $\to_{@}$ can be simulated in $\lambda\mathtt{j/n}$ only in the very special case where t and u both have occurrences of y. In our opinion this is not a limitation: the rule $\to_{@}$ is particularly inefficient since it duplicates even if there is no occurrence of y at all, thus it is rather a good sign that $\lambda\mathtt{j/n}$ cannot simulate $\to_{@}$.

The reduction relation $\lambda\mathtt{j/n}$ does not enjoy PSN, since it is a bit naïve on the way it handles void substitutions. The following counter-example has been found by Stefano Guerrini. Let $u = (zz)[z/y]$, then:

$$\begin{array}{rcllll} t & = & u[x/u] = (zz)[z/y][x/u] & \equiv_{\mathtt{box}_2^0} & (zz)[z/y[x/u]] & \to_{\mathtt{c}} \\ & & (z_1z_2)[z_1/y[x/u]][z_2/y[x/u]] & \to_{\mathtt{d}}^{+} & y[x/u](y[x/u]) & \equiv_{\sigma_2,\mathtt{box}_1^0,\alpha} \\ & & (yy)[x_1/u][x/u] & \equiv_{\mathtt{box}_2^0} & (yy)[x_1/u[x/u]] & \end{array}$$

The term t reduces to a term containing t and so there is a loop of the form $t \to^+ C_0[t] \to^+ C_0[C_1[t]] \to^+ \ldots$. Now, take $t_0 = (\lambda x.((\lambda z.zz)y))((\lambda z.zz)y)$, which is strongly normalizing in the λ-calculus. Since t_0 $\lambda\mathtt{j}/\mathtt{n}$-reduces to t, t_0 is not $\lambda\mathtt{j}/\mathtt{n}$-strongly normalizing and thus $\lambda\mathtt{j}/\mathtt{n}$ does not enjoy PSN. It is worth to note that, in contrast to Melliès counterexample for λ_σ [32], the $\mathtt{dB}$-rule has no role in building the diverging reduction: the fault comes only from the jump subsystem $\mathtt{j}$ modulo $\equiv_{\mathtt{n}}$.

The key point of the previous counter-example is that the jump $[x/u]$ is free to float everywhere in the term since x has no occurrence in t. Such behavior can be avoided by imposing the constraint "$x \in \mathtt{fv}(v)$" to $\mathtt{box}_1^0$ and $\mathtt{box}_2^0$. This has also a natural graphical justification in terms of Pure Proof-Nets ([1], Chapter 6, page 149), since such constraint turns $\mathtt{box}_1^0$ and $\mathtt{box}_2^0$ into the exact analogous of the commutative box-box rule of Linear Logic Proof-Nets, but used here as an equivalence relation. We then modify $\sim_{\mathtt{box}_1^0}$ and $\sim_{\mathtt{box}_2^0}$ by introducing the equivalence $\equiv_{\mathtt{box}}$ as the contextual and reflexive-transitive closure of the equations in Figure 4.

$$\begin{array}{lll} (tv)[x/u] & \sim_{\mathtt{box}_1} & tv[x/u] \qquad \text{if } x \notin \mathtt{fv}(t) \ \&\ x \in \mathtt{fv}(v) \\ t[y/v][x/u] & \sim_{\mathtt{box}_2} & t[y/v[x/u]] \qquad \text{if } x \notin \mathtt{fv}(t) \ \&\ x \in \mathtt{fv}(v) \end{array}$$

Figure 4: The equivalence $\equiv_{\mathtt{box}}$

Now, we redefine $\equiv_{\mathtt{n}}$ in the following way. The **substitution equivalence** $\equiv_{\mathtt{obox}}$ is the smallest equivalence closed by contexts containing all the equations in Figure 5.

$$\begin{array}{lll} t[x/s][y/v] & \sim_{\mathtt{CS}} & t[y/v][x/s] \qquad \text{if } x \notin \mathtt{fv}(v) \ \&\ y \notin \mathtt{fv}(s) \\ \lambda y.(t[x/s]) & \sim_{\sigma_1} & (\lambda y.t)[x/s] \qquad \text{if } y \notin \mathtt{fv}(s) \\ t[x/s]v & \sim_{\sigma_2} & (tv)[x/s] \qquad \text{if } x \notin \mathtt{fv}(v) \\ (tv)[x/u] & \sim_{\mathtt{box}_1} & tv[x/u] \qquad \text{if } x \notin \mathtt{fv}(t) \ \&\ x \in \mathtt{fv}(v) \\ t[y/v][x/u] & \sim_{\mathtt{box}_2} & t[y/v[x/u]] \qquad \text{if } x \notin \mathtt{fv}(t) \ \&\ x \in \mathtt{fv}(v) \end{array}$$

Figure 5: The substitution equivalence $\equiv_{\mathtt{obox}}$

Alternatively, $\equiv_{\mathtt{obox}}$ can be defined by the context closure of the following global permutating equations:

$$\begin{array}{lll} C[\![t[x/u]]\!] & \sim_{\mathtt{obox}} & C[\![t]\!][x/u] \quad \text{if } \mathtt{bs}(C) \cap \mathtt{fv}(u) = \emptyset \text{ and } |t|_x = |H[\![t]\!]|_x > 0 \\ S[\![t[x/u]]\!] & \sim_{\mathtt{obox}} & S[\![t]\!][x/u] \quad \text{if } \mathtt{bs}(S) \cap \mathtt{fv}(u) = \emptyset \text{ and } |t|_x = |S[\![t]\!]|_x = 0 \end{array}$$

where C is any context and S is a spine context.

It is now natural to study $\lambda\mathtt{j}$-reduction modulo $\equiv_{\mathtt{obox}}$. It is easy to prove that the jump calculus terminates with respect to the new equivalence $\equiv_{\mathtt{obox}}$ so that the previous counterexample to PSN is ruled out. We need an auxiliary lemma about potential multiplicities and the $\mathtt{j}$-measure.

Lemma 4.6. *Let $t_0, t_1 \in \mathcal{T}$. If $t_0 \equiv_{\mathtt{obox}} t_1$ then:*
(1) $\mathtt{P}_z(t_0) = \mathtt{P}_z(t_1)$ *for every variable z.*
(2) $\mathtt{jm}(t_0) = \mathtt{jm}(t_1)$.

Proof. By induction on $\equiv_{\mathtt{obox}}$. The base cases are easy calculations, the inductive cases use the i.h. □

Lemma 2.8 and Lemma 4.6:2 together proves the following corollary.

Corollary 4.7. *The reduction relation* $\mathtt{j/obox}$ *is terminating.*

5. Preservation of β-Strong Normalization for $\lambda\mathtt{j/obox}$

The structural λ-calculus modulo $\equiv_{\mathtt{obox}}$ is an incredibly subtle and complex rewriting system, and proving PSN is is not an easy task. Some of the difficulties are:

- *The relation* $\equiv_{\mathtt{obox}}$ *is not a strong bisimulation.* It is not difficult to see that $\lambda\mathtt{j}$ is confluent modulo $\equiv_{\mathtt{obox}}$ (essentially the same proof than for $\lambda\mathtt{j}$). However, $\equiv_{\mathtt{obox}}$ does not preserve reduction lengths to normal form, *i.e.* it is not a strong bisimulation. Two examples can be given by analysing the interaction between $\equiv_{\mathtt{obox}}$ with erasure and duplication. Here is an example for erasure:
$$\begin{array}{ccc} z[x/y][y/u] & \to_{\mathtt{w}} & z[y/u] \\ \equiv_{\mathtt{box}_2} & & \downarrow_{\mathtt{w}} \\ z[x/y[y/u]] & \to_{\mathtt{w}} & z \end{array}$$
and here another one for duplication:
$$\begin{array}{ccccc} (xx)[x/y][y/z] & \to_{\mathtt{c}} & (xx_1)[x/y][x_1/y][y/z] & \to_{\mathtt{c}} & (xx_1)[x/y_1][x_1/y_2][y_1/z][y_2/z] \\ \equiv_{\mathtt{box}_2} & & & & \equiv_{\mathtt{obox}} \\ (xx)[x/y[y/z]] & \to_{\mathtt{c}} & (xx_1)[x/y[y/z]][x_1/y[y/z]] & \equiv_{\alpha} & (xx_1)[x/y_1[y_1/z]][x_1/y_2[y_2/z]] \end{array}$$
Indeed, if $\equiv_{\mathtt{obox}}$ would have been a strong bisimulation, then in both diagrams the two terms of the second column would be $\equiv_{\mathtt{obox}}$-equivalent, while they are not (remark that $\equiv_{\mathtt{obox}}$ preserves the number of constructors so that those terms cannot be $\equiv_{\mathtt{obox}}$-equivalent).
- *The relation* $\equiv_{\mathtt{obox}}$ *cannot be postponed.* The last example shows also that $\equiv_{\mathtt{obox}}$ cannot be postponed. This is illustrated by the upper left corner of the previous figure:
$$\begin{array}{ccc} (xx)[x/y][y/z] & \to_{\mathtt{c}} & (xx_1)[x/y][x_1/y][y/z] \\ \equiv_{\mathtt{box}_2} & & \\ (xx)[x/y[y/z]] & & \end{array}$$
Observe that this phenomenon is caused by the equation $\sim_{\mathtt{box}_2}$. Remark that both composition (*i.e.* $\to_{\mathtt{box}_2}$) and decomposition ($_{\mathtt{box}_2}\!\!\leftarrow$) are used in Guerrini's counterexemple.
- *There is no canonical representant of equivalence classes which is stable by reduction.* Indeed, there are two natural canonical representants in $\lambda\mathtt{j/obox}$. Given t we can define $\mathtt{in}(t)$ as the term obtained by moving all substitution towards the variables as much as possible and $\mathtt{out}(t)$ the term obtained moving substitutions far from variables as much as possible. Consider $t = x[x/(\lambda y.z[z/y])x'] \to_{\mathtt{dB}} x[x/z[z/y][y/x']] = t'$, then $\mathtt{out}(t) = x[x/(\lambda y.z[z/y])x']$ does not reduce to $\mathtt{out}(t') = x[x/z][z/y][y/x']$. Similarly, for the other representative since $\mathtt{in}(t) = (x[y/z]z)[z/z']$ does not reduce to $\mathtt{in}(t') = xz[z/z']$.

In [3] we proved that $\lambda\mathtt{j}$ enjoys PSN in the cases where the equations $\{\sim_{\mathtt{box}_1}, \sim_{\mathtt{box}_2}\}$ are both oriented from left to right or from right to left. Here we prove PSN considering them as equivalences. Surprisingly, the proof of this more general result is relatively more compact and concise than the one(s) in [3]. Indeed, even if we need to pass through an auxiliary calculus, such a calculus can be proved to enjoy PSN without using labels, in contrast to our previous result and proof.

Let us explain our technique. Even if there is no canonical representative form of an $\mathtt{obox}$-equivalence class which is stable by reduction (*cf.* Section 4.2), there is an even more natural way to reason about PSN in the presence of the non-trivial equations $\{\mathtt{box}_1, \mathtt{box}_2\}$ which consists in projecting $\lambda\mathtt{j}/\mathtt{obox}$ over a simpler equational calculus. Since both the calculus and the projection are quite peculiar we introduce them gradually.

A usual naïve idea consists in projecting $\lambda\mathtt{j}/\mathtt{obox}$ into the λ-calculus by means of a function computing the complete unfolding of jumps. This gives the following diagram:

$$\begin{array}{ccc} t & \to_{\lambda\mathtt{j}} & u \\ \downarrow^*_{\mathtt{j}} & & \downarrow^*_{\mathtt{j}} \\ \mathtt{j}(t) & \to^*_\beta & \mathtt{j}(u) \end{array} \tag{5.1}$$

This principle could be easily exploited in order to prove some properties of $\lambda\mathtt{j}/\mathtt{obox}$ (such as confluence), however, this projection erases divergent sub-terms, therefore it cannot be used to prove PSN. For instance, consider $t = x[y/\Omega]$ (where Ω is a non-terminating term), which is only $\lambda\mathtt{j}$-weakly normalizing, whereas $\mathtt{j}(t) = x$ is in normal form. It is easy to show that projection of terms without void jumps preserves divergence and thus PSN. Unfortunately, erasures cannot be postponed in $\lambda\mathtt{j}/\mathtt{obox}$.

Roughly speaking, the projection gives $\mathtt{j}(t) \to^*_\beta \mathtt{j}(u)$ so that there are some steps $t \to_{\lambda\mathtt{j}} u$ s.t. $\mathtt{j}(t) = \mathtt{j}(u)$. It is not really a problem if such (erased) steps are finite, but here there may be infinite sequences of such (erased) steps. It is then quite natural to change the complete unfolding $\mathtt{j}$ into a *non-erasing* unfolding $\mathtt{wj}$, which does not project void jumps:

$$\begin{array}{lcl} \mathtt{wj}(x) & := & x \\ \mathtt{wj}(\lambda x.u) & := & \lambda x.\mathtt{wj}(u) \\ \mathtt{wj}(uv) & := & \mathtt{wj}(u)\mathtt{wj}(v) \\ \mathtt{wj}(t[x/u]) & := & \begin{cases} \mathtt{wj}(t)\{x/\mathtt{wj}(u)\} & \text{if } x \in \mathtt{fv}(t) \\ \mathtt{wj}(t)[x/\mathtt{wj}(u)] & \text{if } x \notin \mathtt{fv}(t) \end{cases} \end{array} \tag{5.2}$$

Note that there are still some erased steps, as for instance $t = x[x/y] \to_{\mathtt{d}} y = u$, where $\mathtt{wj}(t) = y = \mathtt{wj}(u)$, but intuition tells that $\mathtt{wj}$ preserves divergence, because diverging terms are no longer erased by the projection. Note also that the image of the projection of the previous reduction step $t \to_{\mathtt{d}} u$ is no longer a reduction step in the λ-calculus, so that we need to specify which are the rewriting rules and the equations of the image of the translation.

For didactive purpose let us assume that we are able to turn the image of the projection into a calculus — let us say $\lambda\mathtt{void}/\mathtt{o}$ — such that $\mathtt{wj}$ projects $\lambda\mathtt{j}/\mathtt{obox}$ into $\lambda\mathtt{void}/\mathtt{o}$ and preserves divergence. Two important remarks are: since $\mathtt{wj}(\cdot)$ preserves divergence, then PSN for $\lambda\mathtt{void}/\mathtt{o}$ implies PSN for $\lambda\mathtt{j}/\mathtt{obox}$; also, the $\lambda\mathtt{void}/\mathtt{o}$-calculus does not contain the equations $\{\mathtt{box}_1, \mathtt{box}_2\}$ because they were turned into equalities thanks to their side conditions.

It is then reasonable to expect that proving PSN for $\lambda\mathtt{void}/\mathtt{o}$ is easier than PSN for $\lambda\mathtt{j}/\mathtt{obox}$. Our proof technique can then be summarised as follows:

(1) Introduce $\lambda\mathtt{void}/\mathtt{o}$;
(2) Prove PSN for $\lambda\mathtt{void}/\mathtt{o}$;
(3) Show that $\mathtt{wj}(\cdot)$ preserves divergence from $\lambda\mathtt{j}/\mathtt{obox}$ to $\lambda\mathtt{void}/\mathtt{o}$;
(4) Conclude PSN for $\lambda\mathtt{j}/\mathtt{obox}$.

Section 5.1 presents the rewriting rules of $\lambda\mathtt{void/o}$, thus completing point 1. Section 5.2 deals with point 2 and Section 5.3 with points 3 and 4.

We believe that the isolation of $\lambda\mathtt{void/o}$ is an important contribution of this paper. Indeed, it is easy to see that $\lambda\mathtt{void/o}$ should contain at least the three following rewriting rules:

$$\begin{array}{llll} (\lambda x.t)\mathtt{L}\ u & \mapsto_{\beta} & t\{x/u\}\mathtt{L} & \text{if } x \in \mathtt{fv}(t) \\ (\lambda x.t)\mathtt{L}\ u & \mapsto_{\mathtt{dB}} & t[x/u]\mathtt{L} & \text{if } x \notin \mathtt{fv}(t) \\ t[x/u] & \mapsto_{\mathtt{w}} & t & \text{if } x \notin \mathtt{fv}(t) \end{array}$$

More precisely,

- The reduction step $t = (\lambda x.x)y \to_{\mathtt{dB}} x[x/y] = u$ projects into $\mathtt{wj}(t) = (\lambda x.x)\ y \to_{\beta} y = \mathtt{wj}(u)$.
- The reduction step $t = (\lambda x.z)y \to_{\mathtt{dB}} z[x/y] = u$ should map to itself, *i.e.* $\mathtt{wj}(t) = (\lambda x.z)y \to_{\mathtt{dB}} z[x/y] = \mathtt{wj}(u)$.
- The reduction step $t = z[x/y] \to_{\mathtt{w}} z = u$ should map to itself, *i.e.* $\mathtt{wj}(t) = z[x/y] \to_{\mathtt{w}} z = \mathtt{wj}(t)$.

However, projecting on such a simple calculus still does not work. There are three phenomena we should take care of:

(1) **Equations**. As we already mentioned $\equiv_{\mathtt{box}_1}$ and $\equiv_{\mathtt{box}_2}$ vanish, that is, $t \equiv_{\mathtt{box}_1,\mathtt{box}_2} u$ implies $\mathtt{wj}(t) = \mathtt{wj}(u)$. The graphical equivalence, instead, do not vanish, and must be added to the intermediate calculus, thus getting the reduction relation to be considered modulo $\equiv_{\mathtt{o}}$.

(2) **Generalised erasure**. Consider:

$$t = z[x/y_1y_2][y_1/v_1][y_2/v_2] \to_{\mathtt{w}} z[y_1/v_1][y_2/v_2] = u$$

where $\mathtt{wj}(t) = z[x/v_1v_2]$ and $\mathtt{wj}(u) = z[y_1/v_1][y_2/v_2]$. Hence the $\mathtt{w}$-rule $t[x/s] \to t$ if $|t|_x = 0$ must be generalised in order to replace the jump $[x/s]$ by many (eventually none) jumps containing subterms of s. We shall then use the following (Hydra like) rule :

$$t[x/u] \quad \mapsto_{\mathtt{h}} \quad t[x_1/u_1]\dots[x_n/u_n] \qquad \forall i\ (x_i \text{ fresh } \ \&\ u_i \lhd u\ \&\ \mathtt{fv}(u_i) \subseteq \mathtt{fv}(u)\ \&\ n \geq 0)$$

Where $u_i \lhd u$ means that u_i is a subterm of u. The condition upon free variables is necessary in order to avoid unwanted captures inducing degenerated behaviors. Note that the particular case $n = 0$ gives the $\mathtt{w}$-rule.

(3) **Unboxing**: An erasing step $t \to_{\mathtt{w}} u$ can cause jumps to *move* towards the *root* of the term. Consider:

$$t = (zz[x/y])[y/v] \to_{\mathtt{w}} (zz)[y/v] = u$$

where $\mathtt{wj}(t) = zz[x/v]$ and $\mathtt{wj}(u) = (zz)[y/v] =_{\alpha} (zz)[x/v]$. Hence, to project this step over $\lambda\mathtt{void/o}$ we need a rule moving jumps towards the *root* of the term, which could have in principle the general form:

$$C[\![t[x/u]]\!] \to C[\![t]\!][x/u]$$

This rule is the one that shall demand a more involved — but still reasonable — technical development. Indeed, reduction that moves *any* jump towards the root modulo $\equiv_{\mathtt{o}}$ may cause non-termination:

$$\lambda x.x[y/z] \to (\lambda x.x)[y/z] \equiv_{\mathtt{o}} \lambda x.x[y/z] \to \dots$$

In order to avoid this problem we restrict the general form of the rule to a certain kind of contexts, those whose hole is contained in at least one *box*, *i.e.* the argument of an application or the argument of a jump.

We now develop a PSN proof for $\lambda\mathtt{j}/\mathtt{obox}$. Section 5.1 formally defines the intermediate calculus $\lambda\mathtt{void}/\mathtt{o}$, while Section 5.2 proves PSN for the intermediate calculus $\lambda\mathtt{void}/\mathtt{o}$ and Section 5.3 proves the properties of the projection which allows us to conclude PSN for $\lambda\mathtt{j}/\mathtt{obox}$.

Let us conclude this section by observing that the generalised erasure and the unboxing rules are introduced to project the $\mathtt{w}$-rule and not the equations $\{\mathtt{box}_1, \mathtt{box}_2\}$. Said in other words, to prove PSN of the simpler calculus $\lambda\mathtt{j}$ (resp. $\lambda\mathtt{j}/\mathtt{o}$) through the $\mathtt{wj}$ projection into $\lambda\mathtt{void}$ (resp. $\lambda\mathtt{void}/\mathtt{o}$), one still needs the generalised erasure and the unboxing rules. That is why we believe that the technique developed here is really interesting by itself.

5.1. **The $\lambda\mathtt{void}/\mathtt{o}$-calculus.** The $\lambda\mathtt{void}/\mathtt{o}$-calculus can be understood as a memory calculus based on *void* jumps. It is given by a set of terms, written $\mathcal{T}_{\mathtt{v}}$, generated by the following grammar, where only void jumps are allowed:

$$(\mathcal{T}_{\mathtt{v}}) \quad t, u ::= x \mid \lambda x.t \mid tu \mid t[\_/u]$$

The notation $t[\_/u]$ just means that the constant $\_$ has no (free) occurrence in the term t and $\overline{[\_/s]}$ denotes a list of void jumps $[\_/s_1]\dots[\_/s_n]$.

To define the operational semantics we need to define a particular kind of context. More precisely, if C denotes a context then a **boxed context** B is given by the following grammar:

$$B ::= tC \mid t[\_/C] \mid Bt \mid B[\_/t] \mid \lambda y.B$$

We now consider the reduction rules and equations in Figure 6. The notation $\mathtt{L}$ in the rules $\mathtt{dB}$ and β means a list $[\_/u_1]\dots[\_/u_k]$ of void jumps where $k \in \mathbb{N}$ (so potentially $k = 0$).

$$\begin{array}{llll}
(\lambda x.t)\mathtt{L}\ u & \mapsto_{\beta} & t\{x/u\}\mathtt{L} & \text{if } x \in \mathtt{fv}(t) \\
(\lambda x.t)\mathtt{L}\ u & \mapsto_{\mathtt{dB}} & t[_/u]\mathtt{L} & \text{if } x \notin \mathtt{fv}(t) \\
t[_/u] & \mapsto_{\mathtt{h}} & t[_/u_1]\dots[_/u_n] & \forall i\ (u_i \lhd u\ \&\ \mathtt{fv}(u_i) \subseteq \mathtt{fv}(u)\ \&\ n \geq 0) \\
B[\![t[_/u]]\!] & \mapsto_{\mathtt{u}} & B[\![t]\!][_/u] & B \text{ does not bind } u \\
 & & & \\
t[_/s][_/v] & \sim_{\mathtt{CS}} & t[_/v][_/s] & \\
\lambda y.(t[_/s]) & \sim_{\sigma_1} & (\lambda y.t)[_/s] & \text{if } y \notin \mathtt{fv}(s) \\
t[_/s]v & \sim_{\sigma_2} & (tv)[_/s] &
\end{array}$$

Figure 6: The $\lambda\mathtt{void}/\mathtt{o}$-reduction system

Note that the $\mathtt{w}$-rule $t[x/u] \to t$ with $x \notin \mathtt{fv}(t)$ of $\lambda\mathtt{j}$ is a particular case of the $\mathtt{h}$-rule. Remark also that the unboxing rule of $\lambda\mathtt{void}/\mathtt{o}$ moves *void* jumps outside terms, which was forbidden in the equation $\mathtt{box}_2$ of $\lambda\mathtt{j}/\mathtt{obox}$. However, this does not break PSN because there is no *boxing* rule in $\lambda\mathtt{void}/\mathtt{o}$. Indeed, Guerrini's counterexample uses both boxing and unboxing.

We write $t \to_{\lambda\mathtt{void}/\mathtt{o}} t'$ iff $t \equiv_{\mathtt{o}} t_1 \to_{\lambda\mathtt{void}} t_1' \equiv_{\mathtt{o}} t'$, where $\to_{\lambda\mathtt{void}}$ is the reduction relation generated by the previous rewriting rules $\{\beta, \mathtt{dB}, \mathtt{h}, \mathtt{u}\}$ and $\equiv_{\mathtt{o}}$ is the equivalence relation

defined in Figure 2 but restricted here to the $\lambda\mathtt{void}$-syntax. As before, $\to_{\mathcal{R}}$ denotes the contextual closure of $\mapsto_{\mathcal{R}}$, for $\mathcal{R} \subseteq \{\beta, \mathtt{dB}, \mathtt{h}, \mathtt{u}\}$.

We now show some properties of the new memory reduction system which are used in Section 5.2 to show PSN.

Lemma 5.1. *Let $u, v, s \in \mathcal{T}_{\mathtt{v}}$. If $u \lhd s$ and $x \notin \mathtt{fv}(u)$, then $u \lhd s\{x/v\}$.*

Proof. By induction on s. □

Lemma 5.2. *Let $t_0, t_1, u_0, u_1 \in \mathcal{T}_{\mathtt{v}}$.*
- *If $t_0 \to^*_{\mathtt{h,u/o}} t_1$ then $t_0\{x/u_0\} \to^*_{\mathtt{h,u/o}} t_1\{x/u_0\}$.*
- *If $u_0 \to^*_{\mathtt{h,u/o}} u_1$ then $t\{x/u_0\} \to^*_{\mathtt{h,u/o}} t\{x/u_1\}$.*

Proof. Straightforward. □

Lemma 5.3. *Let $t, v, u, s_i \in \mathcal{T}_{\mathtt{v}}$. Let $x \in \mathtt{fv}(v)$. Then $t[\_/v\{x/u\}] \to^*_{\mathtt{h}} t\overline{[\_/s]}[\_/u]$, where $s_i \lhd v$ and $\mathtt{fv}(s_i) \subseteq \mathtt{fv}(v)$ and $x \notin \mathtt{fv}(s_i)$.*

Proof. Straightforward, the case $t[\_/v\{x/u\}] = t\overline{[\_/s]}[\_/u]$ happening in particular when $v = x$ and $\overline{[\_/s]}$ is empty. □

Lemma 5.4. *Let $t, u, v \in \mathcal{T}_{\mathtt{v}}$. If $y \in \mathtt{fv}(t)$ then $t\{y/v[\_/u]\} \to^*_{\mathtt{h,u/o}} t\{y/v\}[\_/u]$.*

Proof. By induction on t. □

Lemma 5.5. *Let $t_0, t_1, v \in \mathcal{T}_{\mathtt{v}}$. If $t_0 \to^+_{\mathtt{h}} t_1$, $x \in \mathtt{fv}(t_0)$ and $x \notin \mathtt{fv}(t_1)$, then $t_0\{x/v\} \to^*_{\mathtt{h,u/o}} t_1[\_/v]$.*

Proof. By induction on the number of steps from t_0 to t_1, and in the base case by induction on the reduction step from t_0 to t_1.

- $t_0 = u_0[\_/u_1] \to_{\mathtt{h}} u_0[\_/v_1]\dots[\_/v_m] = t_1$, where $v_i \lhd u_1$ and $\mathtt{fv}(v_i) \subseteq \mathtt{fv}(u_1)$. Then $x \in u_1$ and $x \notin u_0$ and $x \notin v_i$ so that
$$u_0[\_/u_1]\{x/v\} = u_0[\_/u_1\{x/v\}] \to^*_{\mathtt{h}} \ (\textit{Lem. } 5.3)\ u_0[\_/v_1]\dots[\_/v_m][\_/v]$$
- $t_0 = \lambda y.u_0 \to \lambda y.u_1 = t_1$, where $u_0 \to u_1$. Then,
$$(\lambda y.u_0)\{x/v\} = \lambda y.u_0\{x/v\} \to^*_{\mathtt{h,u/o}} \ (i.h.)\ \lambda y.u_1[\_/v] \equiv_{\sigma_1} (\lambda y.u_1)[\_/v]$$
- $t_0 = u_0 v_0 \to u_1 v_0 = t_1$, where $u_0 \to u_1$. Then,
$$(u_0 v_0)\{x/v\} = u_0\{x/v\}v_0 \to^*_{\mathtt{h,u/o}} \ (i.h.)\ u_1[\_/v]v_0 \equiv_{\sigma_2} (u_1 v_0)[\_/v]$$
- $t_0 = u_0[\_/v_0] \to u_1[\_/v_0] = t_1$, where $u_0 \to u_1$. Then,
$$u_0[\_/v_0]\{x/v\} = u_0\{x/v\}[\_/v_0] \to^*_{\mathtt{h,u/o}} \ (i.h.)\ u_1[\_/v][\_/v_0] \equiv_{\mathtt{cs}} u_1[\_/v_0][\_/v]$$
- All the remaining cases are straightforward. □

Corollary 5.6. *Let* $t_0, t_1, v \in \mathcal{T}_{\mathtt{v}}$. *If* $t_0 \to^+_{\mathtt{h,u/o}} t_1$, $x \in \mathtt{fv}(t_0)$ *and* $x \notin \mathtt{fv}(t_1)$, *then* $t_0\{x/v\} \to^*_{\mathtt{h,u/o}} t_1[\_/v]$.

Proof. By induction on the number of $\mathtt{h}$-steps in the reduction $t_0 \to^+_{\mathtt{h,u/o}} t_1$. Note first that $\to_{\mathtt{u}}$ and $\equiv_{\mathtt{o}}$ do not loose free variables.

- If there is only one $\mathtt{h}$-step, then the reduction is of the form $t \to^*_{\mathtt{u/o}} u_0 \to_{\mathtt{h}} u_1 \to^*_{\mathtt{u/o}} t'$. We have $t\{x/v\} \to^*_{\mathtt{u/o}} u_0\{x/v\} \to^*_{\mathtt{h,u/o}}$ (*Lem.* 5.5) $u_1[\_/v] \to^*_{\mathtt{u/o}} t'[\_/v]$.
- If there are $n > 1$ $\mathtt{h}$-steps, then the reduction is of the form $t_0 \to^*_{\mathtt{u/o}} u_0 \to_{\mathtt{h}} u_1 \to^+_{\mathtt{h,u/o}} t_1$, with $n - 1 < n$ $\mathtt{h}$-steps from u_1 to t_1 we consider two cases.

  If $x \in \mathtt{fv}(u_0) \cap \mathtt{fv}(u_1)$, then x is lost in the subsequence $u_1 \to^+_{\mathtt{h,u/o}} t_1$. We thus have $t_0\{x/v\} \to^*_{\mathtt{u/o}} u_0\{x/v\} \to_{\mathtt{h}} u_1\{x/v\} \to^*_{\mathtt{h,u/o}}$ (*i.h.*) $t_1[\_/v]$.

  If $x \in \mathtt{fv}(u_0) \setminus \mathtt{fv}(u_1)$, then $t_0\{x/v\} \to^*_{\mathtt{u/o}} u_0\{x/v\} \to^*_{\mathtt{h,u/o}} u_1[\_/v]$ (*Lem.* 5.5) $\to^*_{\mathtt{h,u/o}} t_1[\_/v]$. □

5.2. **Preservation of β-Strong Normalization for $\lambda\mathtt{void/o}$.** The proof of PSN for $\lambda\mathtt{void/o}$ we are going to develop in this section is based on the **IE** property (*cf.* Section 3) and follows the main lines of that of Theorem 3.3. Indeed, given $u \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and $t\{x/u\}\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{void}}$ we show that $s = t[\_/u]\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{void}}$ by using a measure on terms which decreases for every one-step $\lambda\mathtt{void/o}$-reduct of s. However, PSN for $\lambda\mathtt{void/o}$ is much more involved: first because of the nature of the reduction rules $\{\mathtt{h}, \mathtt{u}\}$, second because of the equivalence $\equiv_{\mathtt{o}}$.

A first remark is that jumps in $\lambda\mathtt{void/o}$ are all void so in particular they cannot be duplicated. As a consequence, there is no need at first sight to generalise the **IE** property to terms of the form $t[\_/u_i]^1_m\overline{v}^1_n$ as we did before (Theorem 3.3). However, there are now new ways of getting jumps *on the surface* of the term. Indeed, if $t = \lambda y.t'[\_/v]$ and $y \notin \mathtt{fv}(v)$ one has $s = t[\_/u]\overline{v}^1_n \equiv_{\mathtt{o}} (\lambda y.t')[\_/v][\_/u]\overline{v}^1_n$ Things are even more complicated since jumps can also be moved between the arguments $v_1, \ldots, v_n$ as in:

$$s \equiv_{\mathtt{o}} ((\lambda y.t'[\_/v])v_1)[\_/u]\overline{v}^2_n$$

The opposite phenomenon can happen too, *i.e.* the jump $[\_/u]$ can enter inside t, for instance:

$$s \equiv_{\mathtt{o}} \lambda y.(t'[\_/v][\_/u])\overline{v}^1_n$$

The main point is that the measure we shall use to develop the proof of the **IE** property needs to be stable by the equivalence $\equiv_{\mathtt{o}}$, *i.e.* if $s \equiv_{\mathtt{o}} s'$, then s and s' must have this same measure.

In order to handle this phenomenon we are going to split s in two parts: the multiset $\mathbb{SJ}(s)$ of **jumps** of s which *are* or *can* get to the **surface**, and the **trunk** $\mathbb{T}(s)$, *i.e.* the term obtained from s by removing all the jumps in $\mathbb{SJ}(s)$. This *splitting* of the term is then used to generalise the statement of the **IE** as follows:

$$\text{If } \mathbb{T}(s) \in \mathcal{SN}_{\lambda\mathtt{void/o}} \text{ and } u \in \mathcal{SN}_{\lambda\mathtt{void/o}} \text{ for every } [\_/u] \in \mathbb{SJ}(s) \text{ then } s \in \mathcal{SN}_{\lambda\mathtt{void/o}}.$$

An intuition behind the scheme of this proof is that the term $\mathbb{T}(s)$ and the jumps in $\mathbb{SJ}(s)$ are dynamically independent, in the sense that any reduction of s can be seen as an interleaving of a reduction (eventually empty) of $\mathbb{T}(s)$ and reductions (eventually empty) of elements of $\mathbb{SJ}(s)$. Indeed, the void jumps in $\mathbb{SJ}(s)$ *cannot be affected* by a reduction of

$\mathbb{T}(s)$, since none of their free variables is bound in s, and *cannot affect* a reduction of $\mathbb{T}(s)$ since they are void. The unboxing rule slightly complicates things, but morally that is why the new generalised form of the **IE** property holds.

The attentive reader may wonder why we cannot handle the equivalence $\equiv_{\mathtt{o}}$ by using a strong bisimulation argument, as in the case of $\lambda\mathtt{j}/\mathtt{o}$ (*cf.* Theorem 4.3). Unfortunately, $\equiv_{\mathtt{o}}$ is not a strong bisimulation for $\lambda\mathtt{void}$ as the following example shows:

$$\begin{array}{ccc} x[\_/t[\_/x]v] & \to_{\mathtt{h}} & x[\_/t[\_/x]] \\ \equiv_{\mathtt{o}} & & \equiv_{\mathtt{o}} \\ x[\_/(tv)[\_/x]] & \to_{\mathtt{h}} & ? \end{array}$$

Before starting with the technical details of the proof let us add two more important remarks. First, we have just used $\mathbb{T}(s)$ and $\mathbb{SJ}(s)$ for didactic purposes, the actual definitions are parametrised with respect to a set of variables (those which can be captured in the context containing s). Moreover, in order to simplify the proofs we will not work directly with $\mathbb{SJ}(s)$: we are going to define a parametrised predicate $\mathbb{SNSJ}_\Gamma(s)$, which is true when all the jumps in $\mathbb{SJ}(s)$ are in $\mathcal{SN}_{\lambda\mathtt{void}/\mathtt{o}}$, and a parametrised measure $\mathbb{MSJ}_\Gamma(s)$, built out from the elements of $\mathbb{SJ}(s)$. Second, the unboxing rule makes some inductive reasonings non-trivial, so we isolate them in an intermediate lemma (Lemma. 5.11).

Given $s \in \mathcal{T}_{\mathtt{v}}$ and a set of variables Γ, the trunk $\mathbb{T}_\Gamma(s)$ is given by the following inductive definition:

$$\begin{array}{lcll} \mathbb{T}_\Gamma(x) & := & x & \\ \mathbb{T}_\Gamma(tu) & := & \mathbb{T}_\Gamma(t)u & \\ \mathbb{T}_\Gamma(\lambda x.t) & := & \lambda x.\mathbb{T}_{\Gamma\cup\{x\}}(t) & \\ \mathbb{T}_\Gamma(t[\_/u]) & := & \mathbb{T}_\Gamma(t) & \text{if } \mathtt{fv}(u)\cap\Gamma=\emptyset \\ \mathbb{T}_\Gamma(t[\_/u]) & := & \mathbb{T}_\Gamma(t)[\_/u] & \text{otherwise} \end{array}$$

Note that $x \in \mathtt{fv}(s)$ and $x \in \Gamma$ implies $x \in \mathbb{T}_\Gamma(s)$. Next, we define a predicate on $\mathcal{T}_{\mathtt{v}}$ which is true when all surface jumps contain terminating terms:

$$\begin{array}{lcll} \mathbb{SNSJ}_\Gamma(x) & := & \mathit{true} & \\ \mathbb{SNSJ}_\Gamma(tu) & := & \mathbb{SNSJ}_\Gamma(t) & \\ \mathbb{SNSJ}_\Gamma(\lambda x.t) & := & \mathbb{SNSJ}_{\Gamma\cup\{x\}}(t) & \\ \mathbb{SNSJ}_\Gamma(t[\_/u]) & := & \mathbb{SNSJ}_\Gamma(t)\ \&\ u \in \mathcal{SN}_{\lambda\mathtt{void}/\mathtt{o}} & \text{if } \mathtt{fv}(u)\cap\Gamma=\emptyset \\ \mathbb{SNSJ}_\Gamma(t[\_/u]) & := & \mathbb{SNSJ}_\Gamma(t) & \text{otherwise} \end{array}$$

Observe that $s \in \mathcal{SN}_{\lambda\mathtt{void}/\mathtt{o}}$ implies in particular $\mathbb{SNSJ}_\Gamma(s)$ for any set Γ.

For any term $s \in \mathcal{T}_{\mathtt{v}}$ s.t. $\mathbb{SNSJ}_\Gamma(s)$ we define the following multiset measure:

$$\begin{array}{lcll} \mathbb{MSJ}_\Gamma(x) & := & \emptyset & \\ \mathbb{MSJ}_\Gamma(tu) & := & \mathbb{MSJ}_\Gamma(t) \sqcup \mathbb{MSJ}_\Gamma(u) & \\ \mathbb{MSJ}_\Gamma(\lambda x.t) & := & \mathbb{MSJ}_{\Gamma\cup\{x\}}(t) & \\ \mathbb{MSJ}_\Gamma(t[\_/u]) & := & \mathbb{MSJ}_\Gamma(t) \sqcup [\langle \eta_{\lambda\mathtt{void}/\mathtt{o}}(u), |u|\rangle] & \text{if } \mathtt{fv}(u)\cap\Gamma=\emptyset \\ \mathbb{MSJ}_\Gamma(t[\_/u]) & := & \mathbb{MSJ}_\Gamma(t) \sqcup \mathbb{MSJ}_\Gamma(u) & \text{otherwise} \end{array}$$

Now, we can reformulate a generalised statement for the **IE** property on **V**oid jumps as follows:

(**VIE**) For all $t \in \mathcal{T}_{\mathtt{v}}$, if $\mathbb{T}_\emptyset(t) \in \mathcal{SN}_{\lambda\mathtt{void}/\mathtt{o}}$ and $\mathbb{SNSJ}_\emptyset(t)$, then $t \in \mathcal{SN}_{\lambda\mathtt{void}/\mathtt{o}}$.

Some lemmas about basic properties of $\mathbb{T}_\Gamma(t)$, $\mathbb{SNSJ}_\Gamma(t)$ and $\mathbb{MSJ}_\Gamma(t)$ follow.

Lemma 5.7. *Let $t \in \mathcal{T}_{\mathtt{v}}$ and $x \notin \mathtt{fv}(t)$. Then $\mathbb{T}_{\Gamma\cup\{x\}}(t) = \mathbb{T}_{\Gamma}(t)$ and $\mathbb{SNSJ}_{\Gamma\cup\{x\}}(t)$ iff $\mathbb{SNSJ}_{\Gamma}(t)$.*

Proof. Straightforward. □

Lemma 5.8. *Let $t, u \in \mathcal{T}_{\mathtt{v}}$ s.t. $\mathtt{fv}(t) \subseteq \Gamma$. Then,*
(1) $t \to_{\mathtt{h}}^{*} \mathbb{T}_{\Gamma}(t)$.
(2) *If $x \notin \Gamma$ then $\mathbb{T}_{\Gamma\cup\{x\}}(t)\{x/u\} \to_{\mathtt{h}}^{*} \mathbb{T}_{\Gamma}(t\{x/u\})$.*

Proof.
(1) Straightforward induction on t.
(2) By induction on t.
- $t = x$. Then $\mathbb{T}_{\Gamma\cup\{x\}}(t)\{x/u\} = u \to_{\mathtt{h}}^{*}$ (*Point* 1) $\mathbb{T}_{\Gamma}(u) = \mathbb{T}_{\Gamma}(t\{x/u\})$.
- $t = y$. Then $\mathbb{T}_{\Gamma\cup\{x\}}(t)\{x/u\} = y = \mathbb{T}_{\Gamma}(t\{x/u\})$.
- The cases $t = \lambda y.v$ and $t = uv$ are straightforward using the i.h.
- $t = v[\_/w]$. Let us analyse one particular case in detail, the other ones being similar can be proved by application of the definitions and the i.h. Let us suppose $\Gamma \cap \mathtt{fv}(w) = \emptyset$, $x \in \mathtt{fv}(w)$ and $\Gamma \cap \mathtt{fv}(u) = \emptyset$. Then
  $\mathbb{T}_{\Gamma\cup\{x\}}(t)\{x/u\} = \mathbb{T}_{\Gamma\cup\{x\}}(v)[\_/w]\{x/u\} = \mathbb{T}_{\Gamma\cup\{x\}}(v)\{x/u\}[\_/w\{x/u\}]$ and
  $\mathbb{T}_{\Gamma}(t\{x/u\}) = \mathbb{T}_{\Gamma}(v\{x/u\}[\_/w\{x/u\}]) = \mathbb{T}_{\Gamma}(v\{x/u\})$. The i.h. gives
  $\mathbb{T}_{\Gamma\cup\{x\}}(v)\{x/u\} \to_{\mathtt{h}}^{*} \mathbb{T}_{\Gamma}(v\{x/u\})$ and so we conclude with
  $$\mathbb{T}_{\Gamma\cup\{x\}}(v)\{x/u\}[\_/w\{x/u\}] \to_{\mathtt{h}}^{*} \mathbb{T}_{\Gamma}(v\{x/u\})[\_/w\{x/u\}] \to_{\mathtt{h}} \mathbb{T}_{\Gamma}(v\{x/u\})$$ □

Lemma 5.9. *Let $t, u \in \mathcal{T}_{\mathtt{v}}$ and $x \notin \Gamma$. If $\mathbb{SNSJ}_{\Gamma\cup\{x\}}(t)$, $\mathbb{SNSJ}_{\Gamma}(u)$ and $\mathbb{T}_{\Gamma\cup\{x\}}(t)\{x/u\} \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ then $\mathbb{SNSJ}_{\Gamma}(t\{x/u\})$.*

Proof. By induction on t using Lemma 5.7. □

Lemma 5.10. *Let $t_0 \in \mathcal{T}_{\mathtt{v}}$ s.t. $\mathbb{T}_{\Gamma}(t_0) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and $\mathbb{SNSJ}_{\Gamma}(t_0)$. If $t_0 \equiv_{\mathtt{o}} t_1$ then $\mathbb{SNSJ}_{\Gamma}(t_1)$ and $\mathbb{T}_{\Gamma}(t_0) \equiv_{\mathtt{o}} \mathbb{T}_{\Gamma}(t_1)$ and $\mathbb{MSJ}_{\Gamma}(t_0) = \mathbb{MSJ}_{\Gamma}(t_1)$. Thus in particular the equality $\eta_{\lambda\mathtt{void/o}}(\mathbb{T}_{\Gamma}(t_0)) = \eta_{\lambda\mathtt{void/o}}(\mathbb{T}_{\Gamma}(t_1))$ holds.*

Proof. By induction on $t_0 \equiv_{\mathtt{o}} t_1$. □

The next lemma deals with the unboxing rule, which requires a complex induction.

Lemma 5.11. *Let $t_0 \in \mathcal{T}_{\mathtt{v}}$ s.t. $\mathbb{T}_{\Gamma}(t_0) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and $\mathbb{SNSJ}_{\Gamma}(t_0)$. If $t_0 = B[\![s[\_/u]]\!] \to_{\mathtt{u}} B[\![s]\!][\_/u] = t_1$, where B does not bind u, then $\mathbb{SNSJ}_{\Gamma}(t_1)$ and:*
⋆ *If $\Gamma \cap \mathtt{fv}(u) = \emptyset$ then*
  – *Either $\mathbb{T}_{\Gamma}(t_0) = \mathbb{T}_{\Gamma}(t_1)$ and $\mathbb{MSJ}_{\Gamma}(t_0) \sqsupset \mathbb{MSJ}_{\Gamma}(t_1)$,*
  – *Or $\mathbb{T}_{\Gamma}(t_0) \to_{\mathtt{h}} \mathbb{T}_{\Gamma}(t_1)$;*
⋆ *If $\Gamma \cap \mathtt{fv}(u) \neq \emptyset$ then $\mathbb{T}_{\Gamma}(t_0) \to_{\mathtt{u,h/o}}^{+} \mathbb{T}_{\Gamma}(t_1)$.*

Proof. By induction on B.
- Base cases:
  – $B = vC$: then $t_0 = vC[\![s[\_/u]]\!] \to_{\mathtt{u}} (vC[\![s]\!])[\_/u] = t_1$. Hence $\mathbb{T}_{\Gamma}(t_0) = \mathbb{T}_{\Gamma}(v)C[\![s[\_/u]]\!]$ and $\mathbb{SNSJ}_{\Gamma}(t_0)$ iff $\mathbb{SNSJ}_{\Gamma}(v)$. There are two cases:
    (1) $\underline{\Gamma \cap \mathtt{fv}(u) = \emptyset}$: then $\mathbb{T}_{\Gamma}(t_0) \to_{\mathtt{h}} \mathbb{T}_{\Gamma}(v)C[\![s]\!] = \mathbb{T}_{\Gamma}(t_1)$. To show $\mathbb{SNSJ}_{\Gamma}(t_1)$ we need $\underline{\mathbb{SNSJ}_{\Gamma}(v)}$ & $u \in \mathcal{SN}_{\lambda\mathtt{void/o}}$. The first point is equivalent to $\mathbb{SNSJ}_{\Gamma}(t_0)$, which holds by hypothesis, the second holds since u is a subterm of $\mathbb{T}_{\Gamma}(t_0) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$.

(2) $\underline{\Gamma \cap \mathtt{fv}(u) \neq \emptyset}$: then $\mathbb{T}_\Gamma(t_0) \to_{\mathtt{u}} (\mathbb{T}_\Gamma(v)C[\![s]\!])[\_/u] = \mathbb{T}_\Gamma(t_1)$. To show $\mathbb{SNSJ}_\Gamma(t_1)$ we need $\mathbb{SNSJ}_\Gamma(v)$, which holds by hypothesis.

– $B = v[\_/C]$: there are four cases:

(1) $\underline{\Gamma \cap \mathtt{fv}(u) = \emptyset \ \&\ \Gamma \cap \mathtt{fv}(C[\![s]\!]) = \emptyset}$: then $\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(v) = \mathbb{T}_\Gamma(t_1)$. Also, $\mathbb{SNSJ}_\Gamma(t_0)$ so that $\mathbb{SNSJ}_\Gamma(v)$ & $C[\![s[\_/u]]\!] \in \mathcal{SN}_{\lambda\mathtt{void/o}}$. To show $\mathbb{SNSJ}_\Gamma(t_1)$ we need $C[\![s]\!] \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ & $u \in \mathcal{SN}_{\lambda\mathtt{void/o}}$, which clearly follows from $C[\![s[\_/u]]\!] \in \mathcal{SN}_{\lambda\mathtt{void/o}}$. We still need to show that $\mathbb{MSJ}_\Gamma(t_0) \sqsupset \mathbb{MSJ}_\Gamma(t_1)$ which holds because $\mathbb{MSJ}_\Gamma(t_1)$ is just $\mathbb{MSJ}_\Gamma(t_0)$ where the multiset $[\langle \eta_{\lambda\mathtt{void}}(C[\![s[\_/u]]\!]), |C[\![s[\_/u]]\!]|\rangle] \in \mathbb{MSJ}_\Gamma(t_0)$ is replaced by the strictly smaller multiset $[\langle \eta_{\lambda\mathtt{void}}(C[\![s]\!]), |C[\![s]\!]|\rangle, \langle \eta_{\lambda\mathtt{void}}(u), |u|\rangle]$.

(2) $\underline{\Gamma \cap \mathtt{fv}(u) = \emptyset \ \&\ \Gamma \cap \mathtt{fv}(C[\![s]\!]) \neq \emptyset}$: then

$$\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(v)[\_/C[\![s[\_/u]]\!]] \to_{\mathtt{h}} \mathbb{T}_\Gamma(v)[\_/C[\![s]\!]] = \mathbb{T}_\Gamma(t_1)$$

Also, $\mathbb{SNSJ}_\Gamma(t_0)$ implies $\mathbb{SNSJ}_\Gamma(v)$. To show $\mathbb{SNSJ}_\Gamma(t_1)$ we need $\mathbb{SNSJ}_\Gamma(v)$ & $u \in \mathcal{SN}_{\lambda\mathtt{void/o}}$, which then holds by hypothesis and because u is a subterm of $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$.

(3) $\underline{\Gamma \cap \mathtt{fv}(u) \neq \emptyset \ \&\ \Gamma \cap \mathtt{fv}(C[\![s]\!]) = \emptyset}$: then

$$\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(v)[\_/C[\![s[\_/u]]\!]] \to_{\mathtt{u}} \mathbb{T}_\Gamma(v)[\_/C[\![s]\!]][\_/u] \to_{\mathtt{h}} \mathbb{T}_\Gamma(v)[\_/u] = \mathbb{T}_\Gamma(t_1)$$

Also, $\mathbb{SNSJ}_\Gamma(t_0)$ implies $\mathbb{SNSJ}_\Gamma(v)$. To show $\mathbb{SNSJ}_\Gamma(t_1)$ we need $\mathbb{SNSJ}_\Gamma(v)$ & $C[\![s]\!] \in \mathcal{SN}_{\lambda\mathtt{void/o}}$, which holds by the hypothesis and the fact that $C[\![s]\!]$ is a subterm of a $\mathtt{h}$-reduct of $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$.

(4) $\underline{\Gamma \cap \mathtt{fv}(u) \neq \emptyset \ \&\ \Gamma \cap \mathtt{fv}(C[\![s]\!]) \neq \emptyset}$: then

$$\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(v)[\_/C[\![s[\_/u]]\!]] \to_{\mathtt{u}} \mathbb{T}_\Gamma(v)[\_/C[\![s]\!]][\_/u] = \mathbb{T}_\Gamma(t_1)$$

Also $\mathbb{SNSJ}_\Gamma(t_0)$ implies $\mathbb{SNSJ}_\Gamma(v)$, which is equivalent to $\mathbb{SNSJ}_\Gamma(t_1)$.

- Inductive cases:

– $B = B'[\_/v]$: We have $t_0 = B'[\![s[\_/u]]\!][\_/v] \to_{\mathtt{u}} B'[\![s]\!][\_/v][\_/u] = t_1$. Also $B'[\![s[\_/u]]\!] \to_{\mathtt{u}} B'[\![s]\!][\_/u]$ and the hypothesis $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and $\mathbb{SNSJ}_\Gamma(t_0)$ imply in particular $\mathbb{T}_\Gamma(B'[\![s[\_/u]]\!]) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and $\mathbb{SNSJ}_\Gamma(B'[\![s[\_/u]]\!])$. The i.h. gives $\mathbb{SNSJ}_\Gamma(B'[\![s]\!][\_/u])$ and we distinguish several cases:

(1) $\underline{\Gamma \cap \mathtt{fv}(u) = \emptyset \ \&\ \Gamma \cap \mathtt{fv}(v) = \emptyset}$: The hypothesis $\mathbb{SNSJ}_\Gamma(t_0)$ implies in particular $v \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and the i.h. $\mathbb{SNSJ}_\Gamma(B'[\![s]\!][\_/u])$ gives $\mathbb{SNSJ}_\Gamma(B'[\![s]\!])$ & $u \in \mathcal{SN}_{\lambda\mathtt{void/o}}$, so we conclude also $\mathbb{SNSJ}_\Gamma(t_1)$. We now consider two cases:

(a) If $u_0 = \mathbb{T}_\Gamma(B'[\![s[\_/u]]\!]) = \mathbb{T}_\Gamma(B'[\![s]\!][\_/u])$ and $\mathbb{MSJ}_\Gamma(B'[\![s[\_/u]]\!]) \sqsupset \mathbb{MSJ}_\Gamma(B'[\![s]\!]) \sqcup [\langle \eta_{\lambda\mathtt{void/o}}(u), |u|\rangle]$, then $\mathbb{T}_\Gamma(t_0) = u_0 = \mathbb{T}_\Gamma(t_1)$ and $\mathbb{MSJ}_\Gamma(t_0) = \mathbb{MSJ}_\Gamma(B'[\![s[\_/u]]\!]) \sqcup [\langle \eta_{\lambda\mathtt{void/o}}(v), |v|\rangle] \sqsupset \mathbb{MSJ}_\Gamma(B'[\![s]\!]) \sqcup [\langle \eta_{\lambda\mathtt{void/o}}(u), |u|\rangle] \sqcup [\langle \eta_{\lambda\mathtt{void/o}}(v), |v|\rangle] = \mathbb{MSJ}_\Gamma(t_1)$.

(b) If $u_0 = \mathbb{T}_\Gamma(B'[\![s[\_/u]]\!]) \to_{\mathtt{h}} \mathbb{T}_\Gamma(B'[\![s]\!][\_/u]) = u_1$, then $\mathbb{T}_\Gamma(t_0) = u_0 \to_{\mathtt{h}} u_1 = \mathbb{T}_\Gamma(t_1)$.

(2) $\underline{\Gamma \cap \mathtt{fv}(u) = \emptyset \ \&\ \Gamma \cap \mathtt{fv}(v) \neq \emptyset}$: The i.h. $\mathbb{SNSJ}_\Gamma(B'[\![s]\!][\_/u])$ gives $\mathbb{SNSJ}_\Gamma(B'[\![s]\!])$ & $u \in \mathcal{SN}_{\lambda\mathtt{void/o}}$, so we conclude also $\mathbb{SNSJ}_\Gamma(t_1)$. We now consider two cases:

(a) If $u_0 = \mathbb{T}_\Gamma(B'[\![s[\_/u]]\!]) = \mathbb{T}_\Gamma(B'[\![s]\!][\_/u])$ and $\mathbb{MSJ}_\Gamma(B'[\![s[\_/u]]\!]) \sqsupset \mathbb{MSJ}_\Gamma(B'[\![s]\!]) \sqcup [\langle \eta_{\lambda\mathtt{void/o}}(u), |u|\rangle]$, then $\mathbb{T}_\Gamma(t_0) = u_0[\_/v] = \mathbb{T}_\Gamma(B'[\![s]\!])[\_/v] = \mathbb{T}_\Gamma(t_1)$ and $\mathbb{MSJ}_\Gamma(t_0) = \mathbb{MSJ}_\Gamma(B'[\![s[\_/u]]\!]) \sqcup \mathbb{MSJ}_\Gamma(v) \sqsupset \mathbb{MSJ}_\Gamma(B'[\![s]\!]) \sqcup [\langle \eta_{\lambda\mathtt{void/o}}(u), |u|\rangle] \sqcup \mathbb{MSJ}_\Gamma(v) = \mathbb{MSJ}_\Gamma(t_1)$.

(b) If $u_0 = \mathbb{T}_\Gamma(B'[\![s[\_/u]]\!]) \to_{\mathtt{h}} \mathbb{T}_\Gamma(B'[\![s]\!][\_/u]) = u_1$, then

$$\mathbb{T}_\Gamma(t_0) = u_0[\_/v] \to_{\mathtt{h}} u_1[\_/v] = \mathbb{T}_\Gamma(B'[\![s]\!])[\_/v] = \mathbb{T}_\Gamma(t_1)$$

(3) $\underline{\Gamma \cap \mathtt{fv}(u) \neq \emptyset}$: Then the i.h. gives $u_0 = \mathbb{T}_\Gamma(B'\llbracket s[\_/u]\rrbracket) \to^+_{\mathtt{h,u/o}} \mathbb{T}_\Gamma(B'\llbracket s\rrbracket[\_/u]) = u_1$. We consider the following cases.
   (a) $\underline{\Gamma \cap \mathtt{fv}(v) = \emptyset}$: then
   $$\mathbb{T}_\Gamma(t_0) = u_0 \to^+_{\mathtt{h,u/o}} u_1 = \mathbb{T}_\Gamma(B'\llbracket s\rrbracket)[\_/u] = \mathbb{T}_\Gamma(t_1)$$
   Also $\mathbb{SNSJ}_\Gamma(t_0)$ implies $v \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and the i.h. $\mathbb{SNSJ}_\Gamma(B'\llbracket s\rrbracket[\_/u])$ implies $\mathbb{SNSJ}_\Gamma(B'\llbracket s\rrbracket)$, we thus conclude $\mathbb{SNSJ}_\Gamma(t_1)$.
   (b) $\underline{\Gamma \cap \mathtt{fv}(v) \neq \emptyset}$: then
   $$\begin{array}{rcl} \mathbb{T}_\Gamma(t_0) = u_0[\_/v] \to^+_{\mathtt{h,u/o}} u_1[\_/v] & = & \mathbb{T}_\Gamma(B'\llbracket s\rrbracket)[\_/u][\_/v] \equiv_{\mathtt{o}} \\ & & \mathbb{T}_\Gamma(B'\llbracket s\rrbracket)[\_/v][\_/u] = \mathbb{T}_\Gamma(t_1) \end{array}$$
   Also, the i.h. $\mathbb{SNSJ}_\Gamma(B'\llbracket s\rrbracket[\_/u])$ implies $\mathbb{SNSJ}_\Gamma(B'\llbracket s\rrbracket)$, we therefore conclude $\mathbb{SNSJ}_\Gamma(t_1)$.

– The cases $B = \lambda y.B'$ and $B = B'w$ are similar to the previous ones.

□

The following lemma states that the measure we use for proving **VIE** for $\lambda\mathtt{void/o}$ decreases with every rewriting step.

Lemma 5.12. *Let $t_0 \in \mathcal{T}_{\mathtt{v}}$ s.t. $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and $\mathbb{SNSJ}_\Gamma(t_0)$. If $t_0 \to_{\lambda\mathtt{void}} t_1$ then $\mathbb{SNSJ}_\Gamma(t_1)$ and*

– *Either $\mathbb{T}_\Gamma(t_0) \to^+_{\lambda\mathtt{void/o}} \mathbb{T}_\Gamma(t_1)$ or*
– *$\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(t_1)$ and $\mathbb{MSJ}_\Gamma(t_0) \sqsupset \mathbb{MSJ}_\Gamma(t_1)$.*

Proof. By induction on $t_0 \to_{\lambda\mathtt{void}} t_1$.

• Base cases:
– If $t_0 = (\lambda x.s)\mathtt{L}\ u \to_{\mathtt{dB}} s[\_/u]\mathtt{L} = t_1$, where $x \notin \mathtt{fv}(s)$.
  Let $\mathtt{L} := [\_/v_1]\dots[\_/v_k]$, $Q := \{v_i \mid \Gamma \cap \mathtt{fv}(v_i) \neq \emptyset, i \in \{1,\dots,k\}\}$ and $\overline{Q} := \{v_i \mid \Gamma \cap \mathtt{fv}(v_i) = \emptyset, i \in \{1,\dots,k\}\}$. Define $\mathtt{L}_Q$ the sublist of $\mathtt{L}$ containing only the elements in Q. We have
  ⋆ $\mathbb{SNSJ}_\Gamma(t_0)$ iff $\mathbb{SNSJ}_{\Gamma\cup\{x\}}(s) =_{Lem.\ 5.7} \mathbb{SNSJ}_\Gamma(s)$ and $v_j \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ for every $v_j \in \overline{Q}$.
  ⋆ $\mathbb{T}_\Gamma(t_0) = (\lambda x.\mathbb{T}_{\Gamma\cup\{x\}}(s))\mathtt{L}_Q\ u =_{Lem.\ 5.7} (\lambda x.\mathbb{T}_\Gamma(s))\mathtt{L}_Q\ u$.
  There are two cases:
  (1) $\underline{\Gamma \cap \mathtt{fv}(u) \neq \emptyset}$. We have $\mathbb{T}_\Gamma(t_1) = \mathbb{T}_\Gamma(s)[\_/u]\mathtt{L}_Q$. Then $\mathbb{T}_\Gamma(t_0) \to_{\mathtt{dB}} \mathbb{T}_\Gamma(t_1)$. Moreover, $\mathbb{SNSJ}_\Gamma(t_1)$ iff $\mathbb{SNSJ}_\Gamma(s)$ and $v_j \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ for every $v_j \in \overline{Q}$, which holds by the hypothesis $\mathbb{SNSJ}_\Gamma(t_0)$.
  (2) $\underline{\Gamma \cap \mathtt{fv}(u) = \emptyset}$. We have $\mathbb{T}_\Gamma(t_1) = \mathbb{T}_\Gamma(s)\mathtt{L}_Q$. Then $\mathbb{T}_\Gamma(t_0) \to_{\mathtt{dB}} \mathbb{T}_\Gamma(s)[\_/u]\mathtt{L}_Q \to_{\mathtt{h}} \mathbb{T}_\Gamma(s)\mathtt{L}_Q = \mathbb{T}_\Gamma(t_1)$. Moreover, $\mathbb{SNSJ}_\Gamma(t_1)$ iff $\mathbb{SNSJ}_\Gamma(s)$ and $u \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and $v_j \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ for every $v_j \in \overline{Q}$. The first and third parts follow from the hypothesis $\mathbb{SNSJ}_\Gamma(t_0)$ while the second one follows from the hypothesis $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$.
– $t_0 = (\lambda x.s)\mathtt{L}\ u \to_\beta s\{x/u\}\mathtt{L} = t_1$, where $x \in \mathtt{fv}(s)$.
  Let $\mathtt{L}$, Q, $\overline{Q}$ and $\mathtt{L}_Q$ be as in the previous case. We have
  ⋆ $\mathbb{SNSJ}_\Gamma(t_0)$ iff $\mathbb{SNSJ}_{\Gamma\cup\{x\}}(s)$ and $v_j \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ for every $v_j \in \overline{Q}$.
  ⋆ $\mathbb{T}_\Gamma(t_0) = (\lambda x.\mathbb{T}_{\Gamma\cup\{x\}}(s))\mathtt{L}_Q\ u$ with $x \in \mathbb{T}_{\Gamma\cup\{x\}}(s)$.
  Then $\mathbb{T}_\Gamma(t_0) \to_\beta \mathbb{T}_{\Gamma\cup\{x\}}(s)\{x/u\}\mathtt{L}_Q \to^*_{\mathtt{h}\ (Lem.\ 5.8)} \mathbb{T}_\Gamma(s\{x/u\})\mathtt{L}_Q = \mathbb{T}_\Gamma(t_1)$. Thus in particular $\mathbb{T}_{\Gamma\cup\{x\}}(s)\{x/u\} \in \mathcal{SN}_{\lambda\mathtt{void/o}}$.

Since u is a subterm of $\mathbb{T}_\Gamma(t_0)$, then $u \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and so $\mathbb{SNSJ}_\Gamma(u)$. Then $\mathbb{SNSJ}_\Gamma(t_1)$ iff $\mathbb{SNSJ}_\Gamma(s\{x/u\})$ and $v_j \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ for every $v_j \in \overline{Q}$. The first part holds by Lemma 5.9, the second one from the hypothesis $\mathbb{SNSJ}_\Gamma(t_0)$.

– $t_0 = u[\_/v] \to_{\mathtt{h}} u[\_/v_1]\dots[\_/v_k] = t_1$, where $k \geq 0$, $v_j \lhd v$ for all j and $\mathtt{fv}(v_j) \subseteq \mathtt{fv}(v)$. There are two cases:
  (1) $\underline{\Gamma \cap \mathtt{fv}(v) = \emptyset}$: we have that $\mathbb{SNSJ}_\Gamma(t_0)$ implies $\mathbb{SNSJ}_\Gamma(t_1)$. Then $\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(u) = \mathbb{T}_\Gamma(t_1)$, moreover the multiset $[\langle \eta_{\lambda\mathtt{void}}(u), |u|\rangle]$ of $\mathbb{MSJ}_\Gamma(t_0)$ is replaced by the following multiset of $\mathbb{MSJ}_\Gamma(t_1)$: $[\langle \eta_{\lambda\mathtt{void}}(v_1), |v_1|\rangle, \dots, \langle \eta_{\lambda\mathtt{void}}(v_k), |v_k|\rangle]$. Since $\eta_{\lambda\mathtt{void}}(v) \geq \eta_{\lambda\mathtt{void}}(v_i)$ and $|v| > |v_i|$ we thus conclude $\mathbb{MSJ}_\Gamma(t_0) \sqsupset \mathbb{MSJ}_\Gamma(t_1)$.
  (2) $\underline{\Gamma \cap \mathtt{fv}(v) \neq \emptyset}$: let Q and $\overline{Q}$ as in de dB-case. Then $\mathbb{SNSJ}_\Gamma(t_1)$ iff the terms in $\overline{Q}$ are $\mathcal{SN}_{\lambda\mathtt{void/o}}$ and $\mathbb{SNSJ}_\Gamma(u)$ holds: the former requirement holds because $\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(u)[\_/v]$ and so $v \in \mathcal{SN}_{\lambda\mathtt{void/o}}$, the latter because $\mathbb{SNSJ}_\Gamma(t_0)$ iff $\mathbb{SNSJ}_\Gamma(u)$. Last, $\mathbb{T}_\Gamma(t_1) = \mathbb{T}_\Gamma(u)\mathtt{L}_Q$, where $\mathtt{L}_Q$ is the list of substitutions associated to the elements in Q, then
$$\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(u)[\_/v] \to_{\mathtt{h}} \mathbb{T}_\Gamma(u)\mathtt{L}_Q = \mathbb{T}_\Gamma(t_1)$$
– $t_0 = B[\![s[\_/u]]\!] \to_{\mathtt{u}} B[\![s]\!][\_/u] = t_1$. This case holds by Lemma 5.11.

• Inductive cases:
– $t_0 = u[\_/v] \to_{\lambda\mathtt{void}} u[\_/v'] = t_1$ where $v \to_{\lambda\mathtt{void}} v'$. We consider three cases.
  (1) $\underline{\mathtt{fv}(v) \cap \Gamma = \emptyset \ \&\ \mathtt{fv}(v') \cap \Gamma = \emptyset}$: We have $\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(u) = \mathbb{T}_\Gamma(t_1)$. Also $\mathbb{SNSJ}_\Gamma(t_0)$ implies $v \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ so that $v' \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and thus $\mathbb{SNSJ}_\Gamma(t_1)$. Finally,
$$\begin{array}{rll} \mathbb{MSJ}_\Gamma(t_0) = & \mathbb{MSJ}_\Gamma(u) \sqcup [\langle \eta_{\lambda\mathtt{void/o}}(v), |v|\rangle] & \sqsupset \\ & \mathbb{MSJ}_\Gamma(u) \sqcup [\langle \eta_{\lambda\mathtt{void/o}}(v'), |v'|\rangle] & = \mathbb{MSJ}_\Gamma(t_1) \end{array}$$
  (2) $\underline{\mathtt{fv}(v) \cap \Gamma \neq \emptyset \ \&\ \mathtt{fv}(v') \cap \Gamma \neq \emptyset}$: We have $\mathbb{SNSJ}_\Gamma(t_0) = \mathbb{SNSJ}_\Gamma(u) = \mathbb{SNSJ}_\Gamma(t_1)$. Also $\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(u)[\_/v]$ and $\mathbb{T}_\Gamma(t_1) = \mathbb{T}_\Gamma(u)[\_/v']$, thus $\mathbb{T}_\Gamma(t_0) \to^+_{\lambda\mathtt{void/o}} \mathbb{T}_\Gamma(t_1)$.
  (3) $\underline{\mathtt{fv}(v) \cap \Gamma \neq \emptyset \ \&\ \mathtt{fv}(v') \cap \Gamma = \emptyset}$: We have that $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ implies $v \in \mathcal{SN}_{\lambda\mathtt{void/o}}$, so that $v' \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and $\mathbb{SNSJ}_\Gamma(t_1)$.
  Then $\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(u)[\_/v] \to_{\mathtt{h}} \mathbb{T}_\Gamma(u) = \mathbb{T}_\Gamma(t_1)$.
– All the other cases are straightforward. □

Theorem 5.13 (**VIE** for $\lambda\mathtt{void/o}$)**.** *Let $t \in \mathcal{T}_{\mathtt{v}}$ s.t. $\mathbb{T}_\emptyset(t) \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and $\mathbb{SNSJ}_\emptyset(t)$, then $t \in \mathcal{SN}_{\lambda\mathtt{void/o}}$.*

Proof. We proceed by induction on the measure $m(t) = \langle \eta_{\lambda\mathtt{void/o}}(\mathbb{T}_\emptyset(t)), \mathbb{MSJ}_\emptyset(t)\rangle$. To show $t \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ it is sufficient to show $t' \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ for every $\lambda\mathtt{void/o}$-reduct of t. Take any of such reducts t'. Then Lemmas 5.10 and 5.12 guarantee $\mathbb{T}_\emptyset(t') \in \mathcal{SN}_{\lambda\mathtt{void/o}}$ and $\mathbb{SNSJ}_\emptyset(t')$. Moreover, $\equiv_{\mathtt{o}}$ preserves $m(t)$ and $\to_{\lambda\mathtt{void/o}}$ strictly decreases $m(t)$. We thus apply the i.h. to conclude. □

The following is a consequence of the previous theorem: let $t, u, \overline{v}^1_n \in \lambda$-terms and $s = t[\_/u]\overline{v}^1_n$. If $\mathbb{T}_\emptyset(s) = t\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{void}}$ and $\mathbb{SNSJ}_\emptyset(s)$ holds, *i.e.* $u \in \mathcal{SN}_{\lambda\mathtt{void}}$, then $s = t[\_/u]\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{void}}$. Hence:

Corollary 5.14 (**IE** for $\lambda\mathtt{void/o}$)**.** *The $\lambda\mathtt{void/o}$-calculus enjoys the* **IE** *property.*

Corollary 5.15 (PSN for $\lambda\mathtt{void/o}$)**.** *The $\lambda\mathtt{void/o}$-calculus enjoys PSN, i.e. if $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_{\lambda\mathtt{void/o}}$.*

Proof. By Theorem 3.1 it is sufficient to show **F0**, **F1** and **F2**. The first two properties are straightforward. To show **F2** assume $v \in \mathcal{SN}_{\lambda\mathtt{void}}$ and $u\{x/v\}\overline{t}^1_n \in \mathcal{SN}_{\lambda\mathtt{void}}$, both are λ-terms. Then in particular $u, v, \overline{t}^1_n \in \mathcal{SN}_{\lambda\mathtt{void}}$. We show that $t = (\lambda x.u)v\overline{t}^1_n \in \mathcal{SN}_{\lambda\mathtt{void}}$ by induction on $\eta_{\lambda\mathtt{void}}(u) + \eta_{\lambda\mathtt{void}}(v) + \Sigma_i\, \eta_{\lambda\mathtt{void}}(t_i)$. For that, it is sufficient to show that every $\lambda\mathtt{void}$-reduct of t is in $\mathcal{SN}_{\lambda\mathtt{void}}$. If the $\lambda\mathtt{void}$-reduct of t is internal we conclude by the i.h. If $t \to_\beta u\{x/v\}\overline{t}^1_n = t'$ with $x \in \mathtt{fv}(u)$, then $t' \in \mathcal{SN}_{\lambda\mathtt{void}}$ by hypothesis. If $t \to_{\mathtt{dB}} u[\_/v]\overline{t}^1_n = t'$, then $t' \in \mathcal{SN}_{\lambda\mathtt{void}}$ by the **IE** property (Corollary 5.14). There is no other possible $\lambda\mathtt{void}$-reduct of t which is a λ-term and has no jumps. □

5.3. **Projecting $\lambda\mathtt{j}/\mathtt{obox}$ into $\lambda\mathtt{void}/\mathtt{o}$.** In order to relate the $\lambda\mathtt{j}/\mathtt{obox}$ and the $\lambda\mathtt{void}/\mathtt{o}$ calculi we define a projection function from $\lambda\mathtt{j}$-terms to $\lambda\mathtt{void}$-terms:

$$\begin{array}{lcl}
\mathtt{wj}(x) & := & x \\
\mathtt{wj}(\lambda x.t) & := & \lambda x.\mathtt{wj}(t) \\
\mathtt{wj}(tu) & := & \mathtt{wj}(t)\mathtt{wj}(u) \\
\mathtt{wj}(t[x/u]) & := & \begin{cases} \mathtt{wj}(t)\{x/\mathtt{wj}(u)\} & \text{if } x \in \mathtt{fv}(t) \\ \mathtt{wj}(t)[_/\mathtt{wj}(u)] & \text{if } x \notin \mathtt{fv}(t) \end{cases}
\end{array}$$

Notice that $\mathtt{fv}(t) = \mathtt{fv}(\mathtt{wj}(t))$. Also, $\mathtt{wj}(t) = t$ if $t \in \mathcal{T}_\lambda$.

We now state some basic static properties of $\mathtt{wj}$.

Lemma 5.16. *Let $t, u \in \mathcal{T}$. Then, $\mathtt{wj}(t\{x/u\}) = \mathtt{wj}(t)\{x/\mathtt{wj}(u)\}$.*

Proof. By induction on t. □

Lemma 5.17 (Projection)**.** *Let $t_0 \in \mathcal{T}$. Then,*

(1) *$t_0 \to_{\mathtt{dB}} t_1$ implies $\mathtt{wj}(t_0) \to^+_{\beta,\mathtt{dB}} \mathtt{wj}(t_1)$.*
(2) *$t_0 \to_{\mathtt{w},\mathtt{d},\mathtt{c}} t_1$ implies $\mathtt{wj}(t_0) \to^*_{\mathtt{h},\mathtt{u}/\mathtt{o}} \mathtt{wj}(t_1)$.*
(3) *$t_0 \equiv_{\mathtt{o}} t_1$ implies $\mathtt{wj}(t_0) \equiv_{\mathtt{o}} \mathtt{wj}(t_1)$.*
(4) *$t_0 \equiv_{\mathtt{box}_1,\mathtt{box}_2} t_1$ implies $\mathtt{wj}(t_0) = \mathtt{wj}(t_1)$.*

Proof.

- Base cases:
  - $t_0 = (\lambda x.t)\mathtt{L}\ u \to_{\mathtt{dB}} t[x/u]\mathtt{L} = t_1$.
    Let $\mathtt{M} = [\_/\mathtt{wj}(v_i)]^1_m$ (resp. ρ) be the sequence of jumps (resp. the meta-level substitution) resulting from the projection of t_0, *i.e.* $\mathtt{wj}(t_0) = (\lambda x.\mathtt{wj}(t))\mathtt{M}\ \rho\ \mathtt{wj}(u)$.
    If $x \in \mathtt{fv}(t)$, then:
    $$\begin{array}{rcll}
    \mathtt{wj}(t_0) & = & (\lambda x.\mathtt{wj}(t)\rho)[_/\mathtt{wj}(v_i)\rho]^1_m\mathtt{wj}(u) & \to_\beta \\
    & & \mathtt{wj}(t)\rho\{x/\mathtt{wj}(u)\}[_/\mathtt{wj}(v_i)\rho]^1_m & = \\
    & & \mathtt{wj}(t)\{x/\mathtt{wj}(u)\}\rho[_/\mathtt{wj}(v_i)\rho]^1_m & = \mathtt{wj}(t_1)
    \end{array}$$
    If $x \notin \mathtt{fv}(t)$, then:
    $$\begin{array}{rcll}
    \mathtt{wj}(t_0) & = & (\lambda x.\mathtt{wj}(t)\rho)[_/\mathtt{wj}(v_i)\rho]^1_m\mathtt{wj}(u) & \to_{\mathtt{dB}} \\
    & & \mathtt{wj}(t)\rho[_/\mathtt{wj}(u)][_/\mathtt{wj}(v_i)\rho]^1_m & = \\
    & & \mathtt{wj}(t)[_/\mathtt{wj}(u)][_/\mathtt{wj}(v_i)\rho]^1_m\rho & = \quad \mathtt{wj}(t_1)
    \end{array}$$
  - $t_0 = t[x/u] \to_{\mathtt{w}} t = t_1$ where $|t|_x = 0$. Then $\mathtt{wj}(t[x/u]) = \mathtt{wj}(t)[\_/\mathtt{wj}(u)] \to_{\mathtt{h}} \mathtt{wj}(t)$.
  - $t_0 = t[x/u] \to_{\mathtt{d}} t\{x/u\} = t_1$ where $|t|_x = 1$. Then $\mathtt{wj}(t[x/u]) = \mathtt{wj}(t)\{x/\mathtt{wj}(u)\} =_{Lem.\ 5.16} \mathtt{wj}(t\{x/u\})$.

- $t_0 = t[x/u] \to_{\mathtt{c}} t_{[y]_x}[x/u][y/u] = t_1$ where $|t|_x \geq 2$. Then $\mathtt{wj}(t[x/u]) = \mathtt{wj}(t)\{x/\mathtt{wj}(u)\} = \mathtt{wj}(t_{[y]_x})\{y/\mathtt{wj}(u)\}\{x/\mathtt{wj}(u)\} =_{Lem.\ 5.16} \mathtt{wj}(t_{[y]_x}[x/u][y/u])$.
- $t_0 = t[x/u][y/v] \equiv_{\mathtt{CS}} t[y/v][x/u] = t_1$ where $y \notin \mathtt{fv}(u)$ $x \notin \mathtt{fv}(v)$. There are two cases:
  If $x \in \mathtt{fv}(t)$ or $y \in \mathtt{fv}(t)$, then we obtain $\mathtt{wj}(t_0) = \mathtt{wj}(t_1)$.
  If $x \notin \mathtt{fv}(t)$ and $y \notin \mathtt{fv}(t)$, then
  $$\mathtt{wj}(t_0) = \mathtt{wj}(t)[\_/\mathtt{wj}(u)][\_/\mathtt{wj}(v)] \equiv_{\mathtt{CS}} \mathtt{wj}(t)[\_/\mathtt{wj}(v)][\_/\mathtt{wj}(u)] = \mathtt{wj}(t_1)$$
- $t_0 = (\lambda y.t)[x/u] \equiv_{\sigma_1} \lambda y.t[x/u] = t_1$ where $y \notin \mathtt{fv}(u)$. There are two cases:
  If $x \in \mathtt{fv}(\lambda y.t)$, then $\mathtt{wj}(t_0) = (\lambda y.\mathtt{wj}(t))\{x/\mathtt{wj}(u)\} = \lambda y.\mathtt{wj}(t)\{x/\mathtt{wj}(u)\} = \mathtt{wj}(t_1)$.
  If $x \notin \mathtt{fv}(\lambda y.t)$, then $\mathtt{wj}(t_0) = (\lambda y.\mathtt{wj}(t))[\_/\mathtt{wj}(u)] \equiv_{\sigma} \lambda y.\mathtt{wj}(t)[\_/\mathtt{wj}(u)] = \mathtt{wj}(t_1)$.
- $t_0 = (tv)[x/u] \equiv_{\sigma_2} t[x/u]v = t_1$ where $x \notin \mathtt{fv}(v)$.
  There are two cases:
  If $x \in \mathtt{fv}(t)$, then $\mathtt{wj}(t_0) = (\mathtt{wj}(t)\mathtt{wj}(v))\{x/\mathtt{wj}(u)\} = \mathtt{wj}(t)\{x/\mathtt{wj}(u)\}\mathtt{wj}(v) = \mathtt{wj}(t_1)$.
  If $x \notin \mathtt{fv}(t)$, then $\mathtt{wj}(t_0) = (\mathtt{wj}(t)\mathtt{wj}(v))[\_/\mathtt{wj}(u)] \equiv_{\sigma} \mathtt{wj}(t)[\_/\mathtt{wj}(u)]\mathtt{wj}(v) = \mathtt{wj}(t_1)$.
- $t_0 \equiv_{\mathtt{box}_1,\mathtt{box}_2} t_1$. Then trivially $\mathtt{wj}(t_0) = \mathtt{wj}(t_1)$.

• The inductive cases:
- $t_0 = u[x/v] \to$ (resp. $\equiv$) $u'[x/v] = t_1$, where $u \to$ (resp. $\equiv$) u'. If $x \in \mathtt{fv}(u)$ & $x \in \mathtt{fv}(u')$ or $x \notin \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(u')$ then the property is straightforward by the i.h. So let us suppose $x \in \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(u')$ (so that the reduction step is necessarily a $\mathtt{w}$-step). We have $\mathtt{wj}(u) \to^*_{\mathtt{h,u/o}}$ (*i.h.*) $\mathtt{wj}(u')$. But $x \in \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(u')$ implies $x \in \mathtt{fv}(\mathtt{wj}(u))$ & $x \notin \mathtt{fv}(\mathtt{wj}(u'))$ so that in particular we have $\mathtt{wj}(u) \to^+_{\mathtt{h,u/o}} \mathtt{wj}(u')$. Then $\mathtt{wj}(t_0) = \mathtt{wj}(u)\{x/\mathtt{wj}(v)\} \to^*_{\mathtt{h,u/o}} \mathtt{wj}(u')[\_/\mathtt{wj}(v)] = \mathtt{wj}(t_1)$ holds by Corollary 5.6.
- All the other cases are straightforward. □

Here are some interesting examples:

t	$\to$	t'	$\mathtt{wj}(t)$	$\to^*$	$\mathtt{wj}(t')$
$f[y/x][x/u]$	$\to_{\mathtt{w}}$	$f[x/u]$	$f[\_/u]$	$=$	$f[\_/u]$
$f[y/xz][x/u][z/v]$	$\to_{\mathtt{w}}$	$f[x/u][z/v]$	$f[\_/uv]$	$\to_{\mathtt{h}}$	$f[\_/u][\_/v]$
$f[y/xx][x/u]$	$\to_{\mathtt{w}}$	$f[x/u]$	$f[\_/uu]$	$\to^+_{\mathtt{h}}$	$f[\_/u]$
$(f[w/f[y/xz]]g)[x/u][z/v]$	$\to_{\mathtt{w}}$	$(f[w/f]g)[x/u][z/v]$	$(f[\_/f[\_/uv]]g)$	$\to_{\mathtt{h,u}}$	$(f[\_/f]g)[\_/u][\_/v]$

The previous property allows us to conclude with one of the main results of this paper.

Theorem 5.18 (PSN for $\lambda\mathtt{j/obox}$)**.** *The $\lambda\mathtt{j/obox}$-calculus enjoys PSN, i.e. if $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_{\lambda\mathtt{j/obox}}$.*

Proof. We apply Theorem 1.3, where $\mathcal{A} = \lambda\mathtt{void}$, $\mathcal{A}_1 = \{\mathtt{w}, \mathtt{d}, \mathtt{c}\}$, $\mathcal{A}_2 = \{\mathtt{dB}\}$, $\mathtt{E}$ is $\equiv_{\mathtt{obox}}$, $\mathtt{F}$ is $\equiv_{\mathtt{o}}$ and $t\ \mathcal{R}\ \mathtt{wj}(t)$. Property **(P0)** holds by Lemma 5.17:3-4, Property **(P1)** holds by Lemma 5.17:2, Property **(P2)** holds by Lemma 5.17:1 and Property **(P3)** holds by Corollary 4.7. Now, take $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$ so that Corollary 5.15 gives $t \in \mathcal{SN}_{\lambda\mathtt{void/o}}$. Since $\mathtt{wj}(t) = t$, then $t \in \mathcal{SN}_{\lambda\mathtt{j/obox}}$ by application of the Theorem. □

6. Consequences of the main result

In this section we show how the strong result obtained in Section 5.3 can be used to prove PSN for different variants of the $\lambda\mathtt{j/obox}$-calculus.

6.1. **Adding $\{\mathtt{u}\}$ to $\lambda\mathtt{j}/\mathtt{obox}$.** We show that the reduction relation $\mathtt{u}$ of $\lambda\mathtt{void}/\mathtt{o}$ can be added to $\lambda\mathtt{j}/\mathtt{obox}$ without breaking PSN. The main point of this extension is to show that it is safe to consider unboxing (for void jumps) together with the box equations (for non-void jumps). For that, we first extend the rule $\mathtt{u}$ to act on the whole set $\mathcal{T}$ and not only on $\mathcal{T}_{\mathtt{v}}$ (but they still concern void substitutions only). Boxed contexts are extended to non-void jumps as expected, namely:

$$B ::= tC \mid t[x/C] \mid Bt \mid B[x/t] \mid \lambda y.B$$

Then the rule is given by:

$$B[\![t[x/u]]\!] \mapsto_{\mathtt{u}} B[\![t]\!][x/u], \quad \text{where } B \text{ does not bind } u \ \&\ x \notin \mathtt{fv}(t)$$

Indeed, the $\mathtt{wj}$ function maps $\mathtt{u}$-reduction steps of $\{\lambda\mathtt{j},\mathtt{u}\}/\mathtt{obox}$ into $\{\mathtt{h},\mathtt{u}\}$-reduction steps of $\lambda\mathtt{void}/\mathtt{o}$, as the next lemma shows.

Lemma 6.1 (Extended Projection)**.** *Let $t_0 \in \mathcal{T}$. Then, $t_0 \to_{\mathtt{u}} t_1$ implies $\mathtt{wj}(t_0) \to^*_{\mathtt{h},\mathtt{u}/\mathtt{o}} \mathtt{wj}(t_1)$.*

Proof. By induction on the reduction relations.

- $t_0 = B[\![t[x/u]]\!] \to_{\mathtt{u}} B[\![t]\!][x/u] = t_1$ where B does not bind u and $x \notin \mathtt{fv}(t)$. We show a stronger property, namely:

  If $t_0 = C[\![t[x/u]]\!] \to C[\![t]\!][x/u] = t_1$ where C does not bind u and $x \notin \mathtt{fv}(t)$, then $\mathtt{wj}(t_0) \to^*_{\mathtt{h},\mathtt{u}/\mathtt{o}} \mathtt{wj}(t_1)$. Then the property we want show is just a particular case of the stronger property. By α-conversion we assume w.l.g. that x is not even free in $C[\![t]\!]$.

  We reason by induction on C.
  - $t_0 = [\![t[x/u]]\!] \to_{\mathtt{u}} [\![t]\!][x/u] = t_1$. Then $t_0 = t_1$ so that $\mathtt{wj}(t_0) = \mathtt{wj}(t_1)$.
  - $t_0 = C'[\![t[x/u]]\!]v \to_{\mathtt{u}} (C'[\![t]\!]v)[x/u] = t_1$. Then we conclude by using the i.h. and the equivalence $\equiv_{\sigma_2}$.
  - $t_0 = vC'[\![t[x/u]]\!] \to_{\mathtt{u}} (vC'[\![t]\!])[x/u] = t_1$. Then we conclude by using the i.h. and the reduction $\to_{\mathtt{u}}$.
  - $t_0 = \lambda y.C'[\![t[x/u]]\!] \to_{\mathtt{u}} (\lambda y.C'[\![t]\!])[x/u] = t_1$. Then we conclude by using the i.h. and the equivalence $\equiv_{\sigma_1}$.
  - $t_0 = v[y/C'[\![t[x/u]]\!]] \to_{\mathtt{u}} v[y/C'[\![t]\!]][x/u] = t_1$. We reason by cases.

    If $y \notin \mathtt{fv}(v)$, then:

    $$\begin{array}{rcll} \mathtt{wj}(t_0) & = & \mathtt{wj}(v[y/C'[\![t[x/u]]\!]]) & = \\ & & \mathtt{wj}(v)[\_/\mathtt{wj}(C'[\![t[x/u]]\!])] & \to^*_{\mathtt{h},\mathtt{u}/\mathtt{o}} (i.h.) \\ & & \mathtt{wj}(v)[\_/\mathtt{wj}(C'[\![t]\!])[\_/\mathtt{wj}(u)]] & \to_{\mathtt{u}} \\ & & \mathtt{wj}(v)[\_/\mathtt{wj}(C'[\![t]\!])][\_/\mathtt{wj}(u)] & = \mathtt{wj}(t_1) \end{array}$$

    If $y \in \mathtt{fv}(v)$, then:

    $$\begin{array}{rcll} \mathtt{wj}(t_0) & = & \mathtt{wj}(v[y/C'[\![t[x/u]]\!]]) & = \\ & & \mathtt{wj}(v)\{y/\mathtt{wj}(C'[\![t[x/u]]\!])\} & \to^*_{\mathtt{h},\mathtt{u}/\mathtt{o}} (i.h.\ \&\ Lem.\ 5.2) \\ & & \mathtt{wj}(v)\{y/\mathtt{wj}(C'[\![t]\!])[\_/\mathtt{wj}(u)]\} & \to^*_{\mathtt{h},\mathtt{u}/\mathtt{o}} (Lem.\ 5.4) \\ & & \mathtt{wj}(v)\{y/\mathtt{wj}(C'[\![t]\!])\}[\_/\mathtt{wj}(u)] & = \\ & & \mathtt{wj}(v[y/C'[\![t]\!]])[\_/\mathtt{wj}(u)] & = \mathtt{wj}(t_1) \end{array}$$

  - $t_0 = C'[\![t[x/u]]\!][y/v] \to_{\mathtt{u}} C'[\![t]\!][y/v][x/u] = t_1$. Note that $y \notin \mathtt{fv}(u)$, otherwise the rule cannot be applied. We reason by cases.

If $y \notin \mathtt{fv}(C'[\![t]\!])$, then:

$$\begin{array}{rcll} \mathtt{wj}(t_0) & = & \mathtt{wj}(C'[\![t[x/u]]\!][y/v]) & = \\ & & \mathtt{wj}(C'[\![t[x/u]]\!])[\_/\mathtt{wj}(v)] & \to^*_{\mathtt{h,u/o}} (i.h.) \\ & & \mathtt{wj}(C'[\![t]\!])[\_/\mathtt{wj}(u)][\_/\mathtt{wj}(v)] & \equiv_{\mathtt{CS}} \\ & & \mathtt{wj}(C'[\![t]\!])[\_/\mathtt{wj}(v)][\_/\mathtt{wj}(u)] & = \mathtt{wj}(t_1) \end{array}$$

If $y \in \mathtt{fv}(C'[\![t]\!])$, then:

$$\begin{array}{rcll} \mathtt{wj}(t_0) & = & \mathtt{wj}(C'[\![t[x/u]]\!][y/v]) & = \\ & & \mathtt{wj}(C'[\![t[x/u]]\!])\{y/\mathtt{wj}(v)\} & \to^*_{\mathtt{h,u/o}} (i.h.) \\ & & \mathtt{wj}(C'[\![t]\!])[\_/\mathtt{wj}(u)]\{y/\mathtt{wj}(v)\} & = \\ & & \mathtt{wj}(C'[\![t]\!])\{y/\mathtt{wj}(v)\}[\_/\mathtt{wj}(u)] & = \mathtt{wj}(t_1) \end{array}$$

- The inductive cases for the abstraction, the application and reduction inside substitution are straightforward.
- $t_0 = u_0[y/u_1] \to u_0'[y/u_1] = t_1$, where $u_0 \to_{\mathtt{h}} u_0'$ (resp. $u_0 \to_{\mathtt{u}} u_0'$). Since $\mathtt{u}$ preserves free variables, then $y \in \mathtt{fv}(u_0)$ & $y \in \mathtt{fv}(u_0')$ or $y \notin \mathtt{fv}(u_0)$ & $y \notin \mathtt{fv}(u_0')$ so that the property is straightforward by the i.h. □

Theorem 6.2. *The* $\{\lambda\mathtt{j},\mathtt{u}\}/\mathtt{obox}$*-calculus enjoys PSN, i.e.* $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$*, then* $t \in \mathcal{SN}_{\{\lambda\mathtt{j},\mathtt{u}\}/\mathtt{obox}}$.

Proof. We apply Theorem 1.3, where $\mathcal{A} = \lambda\mathtt{void}$, $\mathcal{A}_1 = \{\mathtt{w},\mathtt{d},\mathtt{c},\mathtt{u}\}$, $\mathcal{A}_2 =$ *set*$\mathtt{dB}$, $\mathtt{E}$ is $\equiv_{\mathtt{obox}}$, $\mathtt{F}$ is $\equiv_{\mathtt{o}}$ and $t \mathrel{\mathcal{R}} \mathtt{wj}(t)$. Property **(P0)** holds by Lemma 5.17:3-4, Property **(P1)** holds by Lemmas 5.17:2 and 6.1, Property **(P2)** holds by Lemmas 5.17:1. To show Property **(P3)** we proceed as follows. First of all notice that $\mathtt{u}/\mathtt{obox}$ is trivially terminating, then show that $\mathcal{A}_1/\mathtt{obox}$ is terminating by showing that $t \to_{\mathcal{A}_1/\mathtt{obox}} t'$ implies $\langle \mathtt{jm}(t), t\rangle >_{lex} \langle \mathtt{jm}(t'), t'\rangle$, where the first component of the pair is compared with respect to the multiset order, the second with respect to the terminating relation $\to_{\mathtt{u}/\mathtt{obox}}$, and the stability of $\mathtt{jm}(\_)$ by $\equiv_{\mathtt{obox}}$, which is given by Lemma 4.6:2. Now, take $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$ so that Corollary 5.15 gives $t \in \mathcal{SN}_{\lambda\mathtt{void}/\mathtt{o}}$. Since $\mathtt{wj}(t) = t$, then $t \in \mathcal{SN}_{\{\lambda\mathtt{j},\mathtt{u}\}/\mathtt{obox}}$ by application of the Theorem. □

6.2. **Orienting the axioms of** $\mathtt{obox}$**.** Another interesting result concerns a more traditional form of explicit substitutions calculus, called here the **inner structural** λ**-calculus**, and noted $\lambda\mathtt{j}_{\mathtt{in}}$, whose rules appear in Figure 7.

Let $\to_{\mathtt{in}/\mathtt{CS}}$ be the context closure of the rules $\mapsto_{\mathtt{in}/\mathtt{CS}_{1,2,3,4}}$ modulo $\equiv_{\mathtt{CS}}$.

Lemma 6.3. *The reduction relation* $\to_{\mathtt{in}/\mathtt{CS}}$ *is strongly normalising.*

Proof. Define $M(t)$ to be the sum of all the sizes of the subterms of t directly affected by jumps. It is easily seen that such a measure strictly decreases by one-step rewriting and is invariant by $\equiv_{\mathtt{CS}}$. □

Corollary 6.4. *The inner structural* λ*-calculus* $\lambda\mathtt{j}_{\mathtt{in}}$ *enjoys PSN.*

Proof. By application of Theorem 1.3, where the required properties of the projection of $\lambda\mathtt{j}_{\mathtt{in}}$ into $\lambda\mathtt{j}/\mathtt{obox}$ are guaranteed by Lemmas 5.17 and 6.3. □

$$
\begin{array}{llll}
(\lambda x.t)\mathtt{L}\ u & \mapsto_{\mathtt{dB}} & t[x/u]\mathtt{L} & \\
t[x/u] & \mapsto_{\mathtt{w}} & t & \text{if } |t|_x = 0 \\
t[x/u] & \mapsto_{\mathtt{d}} & t\{x/u\} & \text{if } |t|_x = 1 \\
t[x/u] & \mapsto_{\mathtt{c}} & t_{[y]_x}[x/u][y/u] & \text{if } |t|_x > 1 \\
 & & & \\
(\lambda y.t)[x/u] & \mapsto_{\mathtt{in/cs}_1} & \lambda y.(t[x/u]) & \\
(tv)[x/u] & \mapsto_{\mathtt{in/cs}_2} & t[x/u]v & \text{if } x \notin \mathtt{fv}(v) \\
(tv)[x/u] & \mapsto_{\mathtt{in/cs}_3} & tv[x/u] & \text{if } x \notin \mathtt{fv}(t) \ \&\ x \in \mathtt{fv}(v) \\
t[y/v][x/u] & \mapsto_{\mathtt{in/cs}_4} & t[y/v[x/u]] & \text{if } x \notin \mathtt{fv}(t) \ \&\ x \in \mathtt{fv}(v) \\
 & & & \\
t[x/u][y/v] & \sim_{\mathtt{cs}} & t[y/v][x/u] & \text{if } x \notin \mathtt{fv}(v) \ \&\ y \notin \mathtt{fv}(s)
\end{array}
$$

Figure 7: The inner structural λ-calculus $\lambda\mathtt{j}_{\mathtt{in}}$

$$
\begin{array}{llll}
(\lambda x.t)u & \to_{\mathtt{B}} & t[x/u] & \\
x[x/u] & \to_{\mathtt{d}'} & u & \\
t[x/u] & \to_{\mathtt{w}} & t & \text{if } x \notin \mathtt{fv}(t) \\
(tv)[x/u] & \to_{@_r} & tv[x/u] & \text{if } x \notin \mathtt{fv}(t) \text{ and } x \in \mathtt{fv}(v) \\
(tv)[x/u] & \to_{@_l} & t[x/u]v & \text{if } x \in \mathtt{fv}(t) \text{ and } x \notin \mathtt{fv}(v) \\
(tv)[x/u] & \to_{@} & t[x/u]v[x/u] & \text{if } x \in \mathtt{fv}(t) \text{ and } x \in \mathtt{fv}(v) \\
(\lambda y.t)[x/u] & \to_{\lambda} & \lambda y.t[x/u] & \\
t[x/u][y/v] & \to_{\mathtt{comp}_1} & t[x/u[y/v]] & \text{if } y \notin \mathtt{fv}(t) \text{ and } y \in \mathtt{fv}(u) \\
t[x/u][y/v] & \to_{\mathtt{comp}_2} & t[y/v][x/u[y/v]] & \text{if } y \in \mathtt{fv}(t) \text{ and } y \in \mathtt{fv}(u) \\
 & & & \\
t[x/u][y/v] & \sim_{\mathtt{cs}} & t[y/v][x/u] & \text{if } y \notin \mathtt{fv}(u) \text{ and } x \notin \mathtt{fv}(v) \\
 & & & (\text{and } x \neq y)
\end{array}
$$

Figure 8: The $\lambda\mathtt{es}$-calculus

The inner structural λ-calculus can be seen as a refinement of Kesner's $\lambda\mathtt{es}$ [20], an explicit substitution calculus related to Proof-Nets, whose rules are in Figure 8.

Indeed, only rules $\{@, \mathtt{comp}_2\}$ are not particular cases of rules of $\lambda\mathtt{j}_{\mathtt{in}}$, but they can be decomposed by using duplication followed by propagations as follows:

$$
\begin{array}{ccc}
(tv)[x/u] & \to_{@} & t[x/u]v[x/u] \\
 & & \\
\downarrow_{\mathtt{c}} & & \uparrow^{\mathtt{in/cs}_3} \\
 & & \\
(tv\{x/y\})[x/u][y/u] & \to_{\mathtt{in/cs}_2} & (t[x/u]v\{x/y\})[y/u] \equiv_\alpha (t[x/u]v)[x/u]
\end{array}
$$

It is then straightforward to simulate $\lambda\mathtt{es}$ inside $\lambda\mathtt{j}_{\mathtt{in}}$, so we get:

Corollary 6.5. *The $\lambda\mathtt{es}$-calculus enjoys PSN.*

The second author shows in [21] that from PSN of $\lambda\mathtt{es}$ one can infer PSN of a wide range of calculi, $\lambda\mathtt{x}$, Kesner's $\lambda\mathtt{es}$ and $\lambda\mathtt{esw}$ [20], Milner's calculus $\lambda\mathtt{sub}$ [33], David's and Guillaume's $\lambda_{\mathtt{ws}}$ [9], the calculus with director strings of [41]. Hence PSN for $\lambda\mathtt{j}/\mathtt{obox}$ encompasses most results of PSN in the literature of explicit substitutions.

$$
\begin{array}{llll}
(\lambda x.t)\mathtt{L}\ u & \mapsto_{\mathtt{dB}} & t[x/u]\mathtt{L} & \\
t[x/u] & \mapsto_{\mathtt{w}} & t & \text{if } |t|_x = 0 \\
t[x/u] & \mapsto_{\mathtt{d}} & t\{x/u\} & \text{if } |t|_x = 1 \\
t[x/u] & \mapsto_{\mathtt{c}} & t_{[y]_x}[x/u][y/u] & \text{if } |t|_x > 1 \\
 & & & \\
\lambda y.(t[x/u]) & \mapsto_{\mathtt{out}_1} & (\lambda y.t)[x/u] & \text{if } y \notin \mathtt{fv}(u) \\
t[x/u]v & \mapsto_{\mathtt{out}_2} & (tv)[x/u] & \\
tv[x/u] & \mapsto_{\mathtt{out}_3} & (tv)[x/u] & \\
t[y/v[x/u]] & \mapsto_{\mathtt{out}_4} & t[y/v][x/u] & \\
 & & & \\
t[x/u][y/v] & \sim_{\mathtt{CS}} & t[y/v][x/u] & \text{if } x \notin \mathtt{fv}(v) \ \&\ y \notin \mathtt{fv}(s)
\end{array}
$$

Figure 9: The outer structural λ-calculus $\lambda\mathtt{j}_{\mathtt{out}}$

The interesting feature of $\lambda\mathtt{j}_{\mathtt{in}}$ with respect to $\lambda\mathtt{es}$ is that the propagation subsystem $\to_{\mathtt{in/CS}}$ is not needed in order to compute a normal form. Propagations are rather (linear) re-arrangements of term constructors which may be used as the basis of some term transformations used for compilation or optimisation.

The strength of a splitting of the whole calculus into a core and propagation system lies in the fact that the latter can be changed without affecting the former. In particular, it is possible to orient the axioms $\{\sigma_1, \sigma_2, \mathtt{box}_1, \mathtt{box}_2\}$ in the opposite direction by getting the *outer* structural λ-calculus $\lambda\mathtt{j}_{\mathtt{out}}$, whose rules are in Figure 9.

Observe that in contrast to the inner calculus the outer box rules act also on void jumps, *i.e.* they are not just an orientation of the box equations, but an extension too. This is possible because — as showed earlier (Theorem 6.2) — extending $\lambda\mathtt{j/obox}$ with unboxing for void jumps is safe (while we do not know whether it is safe to extend $\lambda\mathtt{j/obox}$ with boxing for void jumps). Let $\to_{\mathtt{out/CS}}$ be the derived context closure of the outer rules $\mapsto_{\mathtt{out}_{1,2,3,4}}$ modulo $\equiv_{\mathtt{CS}}$.

Lemma 6.6. *The reduction relation $\to_{\mathtt{out/CS}}$ is strongly normalising.*

Corollary 6.7. *The outer structural λ-calculus $\lambda\mathtt{j}_{\mathtt{out}}$ enjoys PSN.*

Proof. By application of Theorem 1.3, where the required properties of the projection of $\lambda\mathtt{j}_{\mathtt{out}}$ into $\lambda\mathtt{j/obox}$ are guaranteed by Lemmas 5.17 and 6.6. □

In fact, it is easily seen that no matter how the axioms $\{\sigma_1, \sigma_2, \mathtt{box}_1, \mathtt{box}_2\}$ are oriented that they get a terminating rewriting system. As for $\lambda\mathtt{j}_{\mathtt{in}}$ and $\lambda\mathtt{j}_{\mathtt{out}}$, PSN can also be proved for the remaining 14 derived calculi, even if it is not clear to what extent they would be interesting.

6.3. **Adding equations to λ-terms.** We briefly present here the results of [4], which extends and complement those of this paper. As discussed in Section 4.1, the equations $\equiv_{\sigma_1}$ and $\equiv_{\sigma_2}$ can be seen as a jump reformulation of Regnier's $\hat{\sigma}$-equivalence on λ-terms after the elimination of dB-redexes. It is also possible to apply the dB-rule in the other sense (*i.e.* as a dB-expansion) to the equations $\{\sim_{\mathtt{box}_1}, \sim_{\mathtt{box}_2}\}$ in order to obtain other equations

on λ-terms. If $x \notin \mathtt{fv}(t)$ and $x \in \mathtt{fv}(v)$, the equation $\sim_{\mathtt{box}_1}$ can be $\mathtt{dB}$-expanded to the new equation $\widehat{\mathtt{box}}$:

$$\begin{array}{ccc} (tv)\,[x/u] & \sim_{\mathtt{box}_1} & tv\,[x/u] \\ \uparrow_{\mathtt{dB}} & & \uparrow_{\mathtt{dB}} \\ (\lambda x.tv)u & \sim_{\widehat{\mathtt{box}}} & t\,((\lambda x.v)u) \end{array}$$

Axiom $\widehat{\mathtt{box}}$ is a more general instance of the rule called $\mathtt{assoc}$ [34, 30, 8] (which usually is not taken modulo but oriented from right to left). The axiom $\sim_{\mathtt{box}_2}$ $\mathtt{dB}$-expands to a special case of $\sim_{\widehat{\mathtt{box}}}$, and thus it is subsumed by it. Indeed:

$$\begin{array}{ccc} t\,[y/v]\,[x/u] & \sim_{\mathtt{box}_2} & t\,[y/v[x/u]] \\ \uparrow_{\mathtt{dB}} & & \uparrow_{\mathtt{dB}} \\ (\,(\lambda y.t)v\,)\,[x/u] & \sim_{\mathtt{box}_1} & (\lambda y.t)v\,[x/u] \\ \uparrow_{\mathtt{dB}} & & \uparrow_{\mathtt{dB}} \\ (\lambda x.((\lambda y.t)v))u & \sim_{\widehat{\mathtt{box}}} & (\lambda y.t)\,((\lambda x.v)u) \end{array}$$

Last, one can turn the unboxing rule into its λ-calculus form, getting:

$$t((\lambda x.v)u) \quad \mapsto_{\hat{\mathtt{u}}} \quad (\lambda x.tv)u \qquad \text{if } x \notin \mathtt{fv}(t) \;\&\; x \in \mathtt{fv}(v)$$

Let $\equiv_{\Pi}$ be defined as the smallest equivalence relation containing $\equiv_{\{\hat{\sigma}_1, \hat{\sigma}_2, \widehat{\mathtt{box}}\}}$ and $\equiv_{\mathtt{obox}}$. In [4] we show that the $\{\lambda\mathtt{j}, \mathtt{u}, \hat{\mathtt{u}}\}/\Pi$-calculus in Figure 10 enjoys PSN. The proof is obtained via a simple function which eliminates $\mathtt{dB}$-redexes, and that project this calculus over the $\{\lambda\mathtt{j}, \mathtt{u}\}/\mathtt{obox}$-calculus, whose PSN is given by Theorem 6.2. The main result of [4], however, is that the the $\{\lambda\mathtt{j}, \mathtt{u}, \hat{\mathtt{u}}\}/\Pi$-calculus is also Church-Rosser modulo the whole equational theory. This is proved via $\mathtt{M}$-developments, a new notion of development taking advantage of jumps. Actually, in [4] we use a macro-steps substitution rule $t[x/u] \to_{\mathtt{sub}} t\{x/u\}$ instead of our subsystem $\to_{\mathtt{j}}$: we do so because the fine granularity of $\to_{\mathtt{j}}$ plays no role in the proof of these properties, their refinement to $\to_{\mathtt{j}}$ is straightforward.

Let us call **permutative λ-calculus** (see Figure 11) the set of λ-terms plus the operational semantics given by $\{\beta, \hat{\mathtt{u}}\}/\mathtt{P}$, where $\equiv_{\mathtt{P}}$ is the smallest equivalence relation containin $\hat{\sigma}_1$, $\hat{\sigma}_2$, $\widehat{\mathtt{box}}$. Such a calculus can be (strictly) simulated into the $\{\lambda\mathtt{j}, \hat{\mathtt{u}}, \mathtt{u}\}/\Pi$-calculus and thus it enjoys PSN. This result generalises all known results in the literature about PSN for λ-calculus extended with permutative conversion [8, 39, 30]. In [4] we also prove that it is Church-Rosser modulo $\equiv_{\mathtt{P}}$.

7. Conclusions

We have introduced the structural $\lambda\mathtt{j}$-calculus, a concise but expressive λ-calculus with jumps admitting graphical interpretations by means of $\lambda\mathtt{j}$-dags and Pure Proof-Nets. Even

$$
\begin{array}{llll}
(\lambda x.t)\mathtt{L}\ u & \mapsto_{\mathtt{dB}} & t[x/u]\mathtt{L} & \\
t[x/u] & \mapsto_{\mathtt{w}} & t & \text{if } |t|_x = 0 \\
t[x/u] & \mapsto_{\mathtt{d}} & t\{x/u\} & \text{if } |t|_x = 1 \\
t[x/u] & \mapsto_{\mathtt{c}} & t_{[y]_x}[x/u][y/u] & \text{if } |t|_x > 1 \\
\\
B[\![t[x/u]]\!] & \mapsto_{\mathtt{u}} & B[\![t]\!][x/u] & B \text{ does not bind } u \\
t((\lambda x.v)u) & \mapsto_{\hat{\mathtt{u}}} & (\lambda x.tv)u & \text{if } x \notin \mathtt{fv}(t) \ \&\ x \notin \mathtt{fv}(v) \\
\\
(\lambda x.\lambda y.t)u & \sim_{\hat{\sigma}_1} & \lambda y.((\lambda x.t)u) & \text{if } y \notin \mathtt{fv}(u) \\
(\lambda x.tv)u & \sim_{\hat{\sigma}_2} & (\lambda x.t)uv & \text{if } x \notin \mathtt{fv}(v) \\
(\lambda x.tv)u & \sim_{\widehat{\mathtt{box}}} & t((\lambda x.v)u) & \text{if } x \notin \mathtt{fv}(t) \ \&\ x \in \mathtt{fv}(v) \\
\\
t[x/s][y/v] & \sim_{\mathtt{CS}} & t[y/v][x/s] & \text{if } x \notin \mathtt{fv}(v) \ \&\ y \notin \mathtt{fv}(s) \\
\lambda y.(t[x/s]) & \sim_{\sigma_1} & (\lambda y.t)[x/s] & \text{if } y \notin \mathtt{fv}(s) \\
t[x/s]v & \sim_{\sigma_2} & (tv)[x/s] & \text{if } x \notin \mathtt{fv}(v) \\
(tv)[x/u] & \sim_{\mathtt{box}_1} & tv[x/u] & \text{if } x \notin \mathtt{fv}(t) \ \&\ x \in \mathtt{fv}(v) \\
t[y/v][x/u] & \sim_{\mathtt{box}_2} & t[y/v[x/u]] & \text{if } x \notin \mathtt{fv}(t) \ \&\ x \in \mathtt{fv}(v)
\end{array}
$$

Figure 10: The structural λ-calculus modulo

$$
\begin{array}{llll}
(\lambda x.t)u & \mapsto_{\beta} & t\{x/u\} & \\
t((\lambda x.v)u) & \mapsto_{\hat{\mathtt{u}}} & (\lambda x.tv)u & \text{if } x \notin \mathtt{fv}(t) \ \&\ x \notin \mathtt{fv}(v) \\
\\
(\lambda x.\lambda y.t)u & \sim_{\hat{\sigma}_1} & \lambda y.((\lambda x.t)u) & \text{if } y \notin \mathtt{fv}(u) \\
(\lambda x.tv)u & \sim_{\hat{\sigma}_2} & (\lambda x.t)uv & \text{if } x \notin \mathtt{fv}(v) \\
(\lambda x.tv)u & \sim_{\widehat{\mathtt{box}}} & t((\lambda x.v)u) & \text{if } x \notin \mathtt{fv}(t) \ \&\ x \in \mathtt{fv}(v)
\end{array}
$$

Figure 11: The permutative λ-calculus

if $\lambda\mathtt{j}$ has strong linear logic background, the calculus can be understood as a particular reduction system, based on the notion of multiplicity and reduction at a distance, and being independent from any logic or type system. We established different properties for $\lambda\mathtt{j}$ such as confluence and PSN. Moreover, full composition holds without any need of structural composition nor commutation of jumps. The $\lambda\mathtt{j}$-calculus admits a graphical operational equivalence $\equiv_{\mathtt{o}}$ allowing to commute jumps with linear constructs. The relation $\equiv_{\mathtt{o}}$ can be naturally understood as Regnier's σ-equivalence on λ-terms and turns out to be a strong bisimulation. Moreover, $\equiv_{\mathtt{o}}$ can be further extended to the substitution equivalence $\equiv_{\mathtt{obox}}$ allowing to commute also jumps and non-linear constructs. The resulting calculus enjoys PSN, a non-trivial result from which one derives several known PSN results.

PSN of $\lambda\mathtt{j}$ modulo $\equiv_{\mathtt{obox}}$ is shown by means of an auxiliary calculus $\lambda\mathtt{void/o}$ which can be understood as a *memory* calculus specified by means of *void* substitutions. A memory calculus due to Klop [27] is often used for termination arguments. Its syntax is usually presented as follows:

$$t, u ::= x \mid \lambda x.t \mid tu \mid [t, u]$$

where $x \in \mathtt{fv}(t)$ for every term $\lambda x.t$ and the memory construct $[t, u]$ is used to collect in u the arguments of the erasing β-redexes. The rule associated to this calculus are:

$$\begin{array}{lll} (\lambda x.t)u & \mapsto_\beta & t\{x/u\} \\ [t, v]u & \mapsto_\pi & [tu, v] \end{array}$$

If one interprets $[t, v]$ as $t[\_/v]$ then Klop's calculus can be mapped into $\lambda\mathtt{void/o}$: β maps to β and π becomes the reduction rule $t[\_/v]u \to (tu)[\_/v]$, which is subsumed by the equation $\equiv_{\sigma_2}$ of $\lambda\mathtt{void/o}$. Indeed, $\lambda\mathtt{void/o}$ is more expressive than Klop's calculus. We claim that $\lambda\mathtt{void/o}$ is interesting on its own and can be used for proving termination results beyond those of this paper.

We do not know whether $\lambda\mathtt{j/obox}$ extended with unrestricted *boxing*, in contrast to $\lambda\mathtt{j/obox}$ extended with unrestricted *unboxing* presented in Section 6.1, enjoys PSN. The point is delicate, indeed from the literature ([32]) we know that unrestricted boxing together with the following traditional explicit substitution rule (without side condition on x):

$$(tv)[x/u] \quad \to_@ \quad t[x/u]v[x/u]$$

break PSN. Now, the rule $\to_@$ cannot be simulated in $\lambda\mathtt{j/obox}$, so it would be interesting to understand if $\lambda\mathtt{j/obox}$ plus unrestricted boxing enjoys PSN.

An interesting research direction would be to formalise the link between $\lambda\mathtt{j}$, linear logic and abstract machines. Indeed, in contrast to explicit substitution calculi, $\lambda\mathtt{j}$ naturally expresses the notion of linear head reduction [7], which relates in a simpler way to Krivine's Abstract Machine [29]. This is because linear head reduction performs the minimal amount of substitutions necessary to find which occurrences of variables will stand in head positions. While this is not a reduction strategy in the usual sense of λ-calculus, it can be seen as a clever way to implement β-reduction by means of proof-nets technology, which can be reformulated in the $\lambda\mathtt{j}$-calculus as a strategy.

The structural λ-calculus has been used in [3] to specify $\mathtt{XL}$-developments, a terminating notion of reduction generalising those of development [17] and superdevelopment [28]. It would be interesting to better understand $\mathtt{XL}$-developments.

It would also be interesting to exploit distance and multiplicities in other frameworks dealing for example with pattern matching, continuations or differential features. A direction which seems particularly challenging is standardization for $\lambda\mathtt{j}$. It would be interesting in particular to obtain a notion of standard reduction which is stable by $\equiv_{\mathtt{o}}$-equivalence (or at least $\equiv_{\mathtt{CS}}$, so that the result would pass to $\lambda\mathtt{j}$-dags). Indeed, classical notions as leftmost-outermost reduction do not easily generalise to $\lambda\mathtt{j}$ modulo $\equiv_{\mathtt{o}}$, where jumps can be swapped and permuted with linear constructors.

Acknowledgements

We would like to thank Stefano Guerrini for stimulating discussions.

References

[1] B. Accattoli. Jumping around the box: graphical and operational studies on Lambda Calculus and Linear Logic. Ph.D. Thesis, Università di Roma La Sapienza, 2011.

[2] B. Accattoli and S. Guerrini. Jumping Boxes. representing Lambda-Calculus Boxes by Jumps. In E. Grädel and R. Kahle, editors, Proc. of 18th Computer Science Logic (CSL), volume 5771 of Lecture Notes in Computer Science, pages 55–70. Springer-Verlag, Sept. 2009.

[3] B. Accattoli and D. Kesner. The structural lambda-calculus. In A. Dawar and H. Veith, editors, Proc. of 24thComputer Science Logic (CSL), volume 6247 of Lecture Notes in Computer Science, pages 381–395. Springer-Verlag, Aug. 2010.
[4] B. Accattoli and D. Kesner. The permutative λ-calculus, 2012. In N. Bjorner and A. Voronkov, editors, Proc. of 18th Int. Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR), volume 7180 of Lecture Notes in Computer Science, pages 23–36. Springer-Verlag, March 2012.
[5] F. Baader and T. Nipkow. Term Rewriting and *All That*. Cambridge University Press, 1998.
[6] R. Bloo and K. Rose. Preservation of strong normalization in named lambda calculi with explicit substitution and garbage collection. In Computing Science in the Netherlands, pages 62–72. NCSRF, 1995.
[7] V. Danos and L. Regnier. Reversible, irreversible and optimal lambda-machines. Theoretical Computer Science, 227(1):79–97, 1999.
[8] R. David. A short proof that adding some permutation rules to preserves sn. Theoretical Computer Science, 412(11):1022–1026, 2011.
[9] R. David and B. Guillaume. A lambda-calculus with explicit weakening and explicit substitution. Mathematical Structures in Computer Science, 11(1):169–206, 2001.
[10] N. G. de Bruijn. Generalizing Automath by Means of a Lambda-Typed Lambda Calculus. In Mathematical Logic and Theoretical Computer Science, number 106 in Lecture Notes in Pure and Applied Mathematics, pages 71–92. Marcel Dekker, 1987.
[11] R. Di Cosmo, D. Kesner, and E. Polonovski. Proof nets and explicit substitutions. Mathematical Structures in Computer Science, 13(3):409–450, 2003.
[12] J. Espírito Santo. A note on preservation of strong normalisation in the λ-calculus. Theoretical Computer Science, 412(12-14):169–183, 2011.
[13] J.-Y. Girard. Linear logic. Theoretical Computer Science, 50, 1987.
[14] J.-Y. Girard. Geometry of interaction I: an interpretation of system F. Proc. of the Logic Colloquium, 88:221–260, 1989.
[15] M. Hasegawa. Models of Sharing Graphs: A Categorical Semantics of let and letrec, volume Distinguished Dissertation Series. Springer-Verlag, 1999.
[16] H. Herbelin and S. Zimmermann. An operational account of call-by-value minimal and classical lambda-calculus in "natural deduction" form. In P.-L. Curien, editor, Proc. of 9th Typed Lambda Calculus and Applications (TLCA), volume 5608 of Lecture Notes in Computer Science, pages 142–156. Springer-Verlag, July 2009.
[17] J. R. Hindley. Reductions of residuals are finite. Transactions of the American Mathematical Society, 240:345–361, 1978.
[18] G. Huet. Résolution d'équations dans les langages d'ordre 1, 2, . . . , ? Thèse de doctorat d'état, Université Paris VII, 1976.
[19] F. Kamareddine. Postponement, conservation and preservation of strong normalization for generalized reduction. Journal of Logic and Computation, 10(5):721–738, 2000.
[20] D. Kesner. The theory of calculi with explicit substitutions revisited. In J. Duparc and T. A. Henzinger, editors, Proc. of 16th Computer Science Logic (CSL), volume 4646 of Lecture Notes in Computer Science, pages 238–252. Springer-Verlag, Sept. 2007.
[21] D. Kesner. A theory of explicit substitutions with safe and full composition. Logical Methods in Computer Science, 5(3:1):1–29, 2009.
[22] D. Kesner and S. O. Conchúir. Milner's lambda calculus with partial substitutions, 2008.
[23] D. Kesner and S. Lengrand. Extending the explicit substitution paradigm. In J. Giesl, editor, 16th International Conference on Rewriting Techniques and Applications (RTA), volume 3467 of Lecture Notes in Computer Science, pages 407–422. Springer-Verlag, Apr. 2005.
[24] D. Kesner and S. Lengrand. Resource operators for lambda-calculus. Information and Computation, 205(4):419–473, 2007.
[25] D. Kesner and F. Renaud. The prismoid of resources. In R. Královic and D. Niwinski, editors, Proc. of the 34th Mathematical Foundations in Computer Science, volume 5734 of Lecture Notes in Computer Science, pages 464–476. Springer-Verlag, Aug. 2009.
[26] A. J. Kfoury and J. B. Wells. New notions of reduction and non-semantic proofs of beta-strong normalization in typed lambda-calculi. In D. Kozen, editor, 10th Annual IEEE Symposium on Logic in Computer Science (LICS), pages 311–321. IEEE Computer Society Press, June 1995.

[27] J.-W. Klop. Combinatory Reduction Systems, volume 127 of Mathematical Centre Tracts. Mathematisch Centrum, Amsterdam, 1980. PhD Thesis.
[28] J.-W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. Theoretical Computer Science, 121(1/2):279–308, 1993.
[29] J.-L. Krivine. Un interpréteur du lambda-calcul. Available on `http://www.pps.jussieu.fr/~krivine/articles/`.
[30] S. Lengrand. Termination of lambda-calculus with the extra call-by-value rule known as assoc. CoRR, abs/0806.4859, 2008.
[31] J. Maraist, M. Odersky, D. N. Turner, and P. Wadler. Call-by-name, call-by-value, call-by-need and the linear lambda calculus. Theoretical Computer Science, 228(1-2):175–210, 1999.
[32] P.-A. Melliès. Typed lambda-calculi with explicit substitutions may not terminate. In M. Dezani-Ciancaglini and G. D. Plotkin, editors, Proc. of 2nd Typed Lambda Calculus and Applications (TLCA), volume 902 of Lecture Notes in Computer Science, pages 328–334. Springer-Verlag, Apr. 1995.
[33] R. Milner. Local Bigraphs and Confluence: Two Conjectures (extended abstract). Electronic Notes in Theoretical in Computer Science, 175(3):65–73, 2007.
[34] E. Moggi. Computational lambda-calculus and monads. In R. Parikh, editor, 4th Annual IEEE Symposium on Logic in Computer Science (LICS), pages 14–23. IEEE Computer Society Press, June 1989.
[35] R. P. Nederpelt. The fine-structure of lambda calculus. Technical Report CSN 92/07, Eindhoven Univ. of Technology, 1992.
[36] Y. Ohta and M. Hasegawa. A terminating and confluent linear lambda calculus. In F. Pfenning, editor, Rewriting Techniques and Applications (RTA), volume 4098 of Lecture Notes in Computer Science, pages 166–180. Springer-Verlag, 2006.
[37] L. Regnier. Une équivalence sur les lambda-termes. Theoretical Computer Science, 2(126):281–292, 1994.
[38] F. Renaud. Les ressources explicites vues par la théorie de la réécriture. Ph.D. Thesis, Université Paris-Diderot, 2011.
[39] J. E. Santo. A note on preservation of strong normalisation in the -calculus. Theoretical Computer Science, 412(11):1027–1032, 2011.
[40] H. Schwichtenberg. Termination of permutative conversions in intuitionistic Gentzen calculi. Theoretical Computer Science, 212(1-2):247–260, 99.
[41] F.-R. Sinot, M. Fernández, and I. Mackie. Efficient reductions with director strings. In R. Nieuwenhuis, editor, 14th International Conference on Rewriting Techniques and Applications (RTA), volume 2706 of Lecture Notes in Computer Science, pages 46–60. Springer-Verlag, June 2003.
[42] Terese. Term Rewriting Systems, volume 55 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003.
[43] N. Yoshida. Optimal reduction in weak-lambda-calculus with shared environments. In Proc. of Int. Conference on Functional Programming Languages and Computer Architecture, pages 243–252. ACM Press, June 1993.